Curve-Skeleton Extraction Using Iterative Least Squares Optimization

Yu-Shuen Wang and Tong-Yee Lee, *Member*, *IEEE*

Abstract—A curve skeleton is a compact representation of 3D objects and has numerous applications. It can be used to describe an object's geometry and topology. In this paper, we introduce a novel approach for computing curve skeletons for volumetric representations of the input models. Our algorithm consists of three major steps: 1) using iterative least squares optimization to shrink models and, at the same time, preserving their geometries and topologies, 2) extracting curve skeletons through the thinning algorithm, and 3) pruning unnecessary branches based on shrinking ratios. The proposed method is less sensitive to noise on the surface of models and can generate smoother skeletons. In addition, our shrinking algorithm requires little computation, since the optimization system can be factorized and stored in the precomputational step. We demonstrate several extracted skeletons that help evaluate our algorithm. We also experimentally compare the proposed method with other well-known methods. Experimental results show advantages when using our method over other techniques.

Index Terms—Curve skeletons, shrinking, iterative least squares optimization, thinning, branch pruning.

1 Introduction

Skeleton extraction plays a very important role in many applications, including virtual colonoscopies and virtual endoscopies [19], [20], collision detection [28], computer animation [6], [17], [18], [23], [25], [26], [29], [30], [39], [41], 3D object registration and visualization [3], [4], [33], surface reconstruction [27], shape matching [9], [34], [35], vessel tracking [3], and curved planar reformation [21], [22]. There are several properties or requests for skeletons because there are different requirements for different applications. These properties include homotopy, connectivity, hierarchy, model reconstruction, thinness, centeredness, robustness, reliability, smoothness, etc. However, some of these properties conflict with one another, for example, thinness and model reconstruction. It is difficult to consider these two properties at the same time. Therefore, researchers have developed various algorithms to extract skeletons with different properties for different requirements.

There have been many methods developed for computing skeletons since they can be used in a broad array of areas. However, most of these methods are sensitive to noise or are computationally expensive when used to obtain results. Both shortcomings cannot be simultaneously avoided. In this paper, we introduce a novel algorithm for extracting curve skeletons with the following two advantages: they are 1) computational efficiency and 2) noise insensitivity. Our key idea is to shrink the volumetric model into a thinner one, which allows for a better shape when extracting a skeleton. To implement this idea, we shorten the distances between adjacent voxels and simultaneously constrain the boundary voxels close to their original positions. We then apply a traditional thinning algorithm [32] to remove voxels from the model's boundary when obtaining a 1D skeleton. The extracted skeleton is smooth and close to the model's center, since the voxels are moved during the shrinking process.

• *The authors are with the Computer Graphics Group/Visual System Laboratory, Department of Computer Science and Information Engineering, National Cheng-Kung University, No. 1, Ta-Hsueh Road, Tainan, 701, Taiwan, R.O.C.*
E-mail: braveheart@csie.ncku.edu.tw, tonylee@ncku.edu.tw.

Manuscript received 14 July 2007; revised 5 Nov. 2007; accepted 5 Feb. 2008; published online 19 Feb. 2008.
Recommended for acceptance by M.-P. Cani.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2007-07-0084.
Digital Object Identifier no. 10.1109/TVCG.2008.38.

2 Related Work

There have been many skeleton extraction algorithms proposed for 2D shapes and 3D models. In 2D, skeletons are represented by a medial axis. However, in 3D, some algorithms extract medial surfaces, and some extract one-voxel thick curve skeletons. Formally, the skeleton is a set of centers of maximal inscribed balls that can be either a medial axis or surface. Let $\mathbf{X} \subset \Re^3$ be a 3D object, and the skeleton $\mathbf{Sk} \subset \mathbf{X}$ represents the centers of those inscribed balls. However, this kind of skeleton is not adequate for all kinds of applications. As mentioned earlier, a skeleton has many different properties due to different requirements. Therefore, it is difficult to give a precise definition for a curve skeleton. In addition, many methods have been proposed that deal with different types of models such as polyhedral models [14], [23], [24], [28], [37], volumetric models [5], [7], [10], [13], [17], [18], [19], [20], [31], [40], and point cloud sets [1], [8], [27], [36]. These algorithms can be generally classified into four groups: thinning, distance field, geometric methods, and general field functions. These classifications are briefly described in the following paragraphs. For more details, please refer to [12] and [27].

Among these four classifications, the thinning methods are very efficient. These methods remove the voxels from the model's boundary until there is no qualified voxel that can be removed. By applying masks to test if the boundary voxels can be removed or not, thinning algorithms are able to preserve the model's topology. However, the extracted skeletons are not always smooth by thinning algorithms

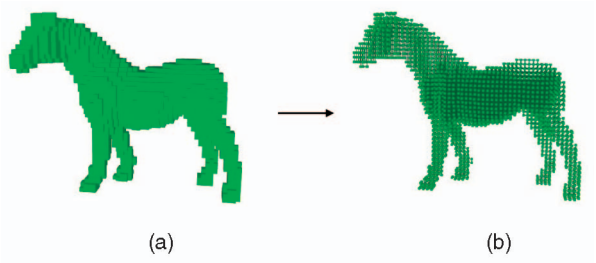

Fig. 1. The volumetric model is considered as a graph, and the 6-adjacency relationship is determined.

and are sensitive to surface perturbation. Distance field algorithms can also efficiently extract skeletons. They first determine the distance for each voxel to its nearest boundary and then obtain the skeleton from those ridges. With results similar to thinning methods, the extracted skeletons are not always smooth and are also sensitive to noise.

In order to compute smooth skeletons from various models, some researchers propose potential field algorithms, which also compute a value for each voxel. When determining the field, they consider larger boundary areas, rather than just the closest boundary voxel. This strategy avoids tiny differences on the model's surface, and therefore, the extracted skeletons are smooth and insensitive to noise. However, the algorithms suffer from heavy computation since determining the field of an interior voxel takes into account many boundary voxels, which may result in an $O(n^2)$ computation cost, where n represents the number of voxels.

Some methods extract skeletons by computing the medial axis. However, there is a common problem with these algorithms. They do not directly extract thin skeletons from 3D models because a large portion of the medial surface has the same distance to the boundary. Therefore, the postprocessing methods are needed. For instance, [14] computes the medial geodesic function when defining the center of a medial surface and erodes the faces to obtain a thin skeleton. Their method extracts very good skeletons from polyhedral models. However, the method suffers from heavy computation since the shortest paths algorithm is used to determine the geodesic distance.

In this paper, a volumetric model is considered as a graph. There is also a connected edge between each 6-adjacent voxel pair. By minimizing our proposed energy function, we transform the graph into a thinner version, which briefly represents the model's shape. By applying the thinning algorithm, the voxels on the shrunk model are then systematically removed to extract a 1D skeleton. Our algorithm costs low computation because solving the objective function and iteratively removing the boundary voxels are both efficient. We utilize the efficiency and homotopy properties of the thinning method but enhance its robustness against noise and the smoothness of the extracted skeletons by using this shrinking algorithm.

3 Algorithm

3.1 Overview

Our algorithm computes the skeletons from volumetric models. The polyhedral models used in this paper are first voxelized by using [16]. In addition, we consider the two voxels connected if they share the same face (i.e., 6-adjacency). In other words, the volumetric model is represented as a graph, $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$, with nodes $\mathbf{V}$ and edges $\mathbf{E}$, where $\mathbf{V} = [\mathbf{v}_1^T, \mathbf{v}_2^T, \ldots, \mathbf{v}_k^T]$ and $\mathbf{v}_i \in \Re^3$ denotes the position of the voxel center (Fig. 1). We classify the voxels into two groups, one of which consists of the boundary voxels $\partial\mathbf{V}$ and the other consisting of interior voxels $\mathbf{V} - \partial\mathbf{V}$. The interior voxels are inside the model, which have six neighbors, while the boundary voxels are located on the model's surface. Initially, the edge lengths connecting the neighboring voxels are the same (Fig. 2a). To transform the original model $\mathbf{G}$ to the shrunk model $\mathbf{G}'$, our algorithm shortens the connected edges, as well as more closely retains the boundary voxels $\partial\mathbf{V}$ to their original positions (Fig. 2b). To implement this idea, we formulate the shrinking process as an objective function and iteratively update voxel positions. Thereafter, the thinning algorithm [32] is applied to remove the boundary voxels of $\mathbf{G}'$; the 1D skeleton $\mathbf{Sk} \subset \mathbf{G}'$ is then obtained. In contrast to the skeleton computed by directly thinning, our extracted skeleton is smoother and closer to the model's center (Fig. 2d).

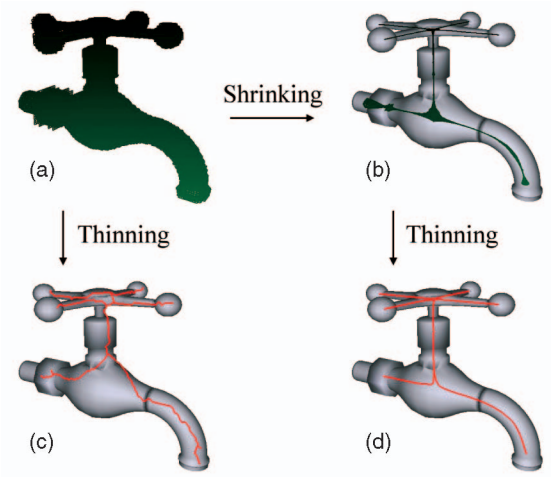

Fig. 2. System overview. (a) The original volumetric model. (b) The shrunk model transformed by our algorithm. (c) By applying the thinning method to the original model, the extracted skeleton is jagged and deviates from the model's center. (d) By applying the thinning method to our shrunk model, a smooth and centered skeleton is obtained.

3.2 Shrinking

Obviously, a skeleton is a 1D representation containing zero volume. Based on this criterion, our goal is to extract the model's skeleton by reducing its volume. Our method shrinks a model by contracting the edges between adjacent voxels. However, homogeneously shortening edge lengths results in only a smaller version of the original model and repeating the same method merely contracts the model into a point. To preserve the model's geometry, we add forces that pull the boundary voxels back to their original positions, denoted as boundary constraints. These forces can reduce the movement of boundary voxels and therefore preserve the model's features. Specifically, we illustrate this idea in Fig. 3c. The shrunk model is represented in green, and the forces that constrain the boundary voxels are represented with blue lines. In this example, it is clear to see that most of the forces pull the boundary voxels upward or

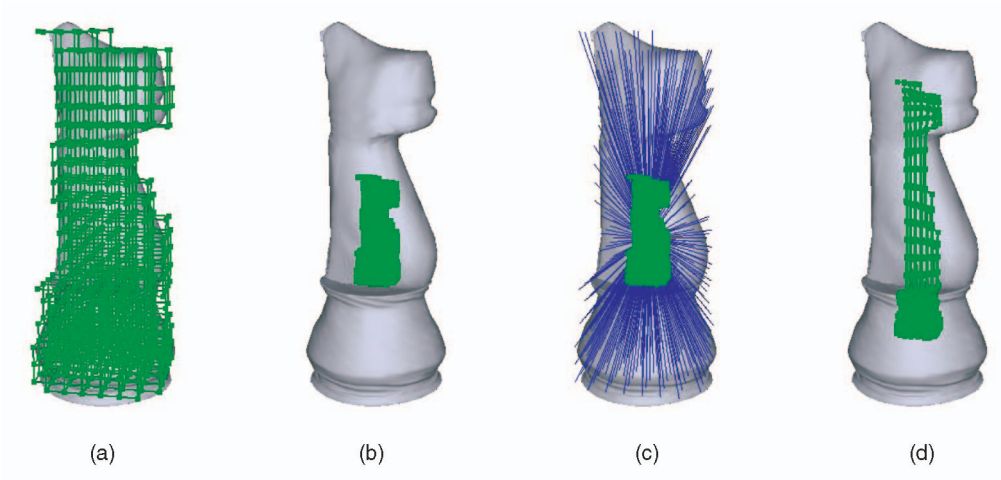

Fig. 3. (a) The original model. (b) We shrink the model (green) without applying boundary constraints. (c) In this case, the boundary forces (blue lines) try to pull the voxels back to their original positions. (d) The geometry is therefore preserved by combining the contraction of edge lengths and boundary constraints.

downward. Thus, the resultant forces prevent the model from shrinking in a vertical direction. In other words, with the aid of the boundary constraints, we can shrink a model nonhomogeneously and transform the model into a thinner version (Fig. 3d). We repeat this manner when shrinking a model, which can possibly prevent features from missing. To implement this idea, we reduce the volume of model $\mathbf{G}$ in an iterative manner, which repeatedly updates the voxel positions from $\mathbf{V}^t$ to $\mathbf{V}^{t+1}$ (t is the iteration number). Our shrinking algorithm is formulated as an objective function, which consists of two energy terms: 1) the boundary constraint E_B and 2) the edge contraction E_Q.

3.2.1 Boundary Constraint

To preserve the model's geometry, the shapes of $\mathbf{G}$ and $\mathbf{G}'$ should be similar. Therefore, we constrain the movement of boundary voxels, which forces the voxels to remain closer to their original positions. Specifically, we minimize the energy term:

$$E_B = \sum |\mathbf{v}_b^{t+1} - \mathbf{v}_b^t|^2 \quad \mathbf{v}_b \in \partial\mathbf{V}. \tag{1}$$

3.2.2 Edge Contraction

We attempt to shorten the edge lengths to shrink a model. Obviously, the contraction amounts for different edges vary since our goal is to shrink the model nonhomogeneously. In other words, some edges should be compressed more, but some should be less. However, we have no idea how to decide the contraction amounts for different directions. Fortunately, our boundary constraints help us achieve this goal because the lengths of some edges should be longer than others in order to resist a significant change in the model's shape. Thus, we introduce the energy:

$$E_Q = \sum_{\{i,j\}\in\mathbf{E}} \left| \left(\mathbf{v}_i^{t+1} - \mathbf{v}_j^{t+1}\right) - \left(p_{ij}^t\right)^2 \left(\mathbf{v}_i^t - \mathbf{v}_j^t\right) \right|^2 \quad \mathbf{v} \in \mathbf{V}, \tag{2}$$

where $p_{ij}^t = l_{ij}^t / l_{ij}^{t-1}$ is the contraction ratio, and l_{ij}^t and l_{ij}^{t-1} are the lengths of $v_i^t - v_j^t$ and $v_i^{t-1} - v_j^{t-1}$, respectively. In the beginning, we set $p_{ij}^0 = 0$ and apply only E_B to preserve the model's geometry. Since the boundary constraints limit the movement of some voxels, we can therefore obtain the contraction ratios of edges after the first iteration. We also square the value of p_{ij}^t to let the edge contract more if its contraction ratio is smaller (i.e., larger shrinkage) than others. Note that it is possible for p_{ij}^t to become larger than 1.0 during the shrinking process. In this case, we set $p_{ij}^t = 1.0$. Otherwise, the edge will become much longer in the next iteration.

3.2.3 Solving Optimization

By combining the two energy terms, we compute the shrunk model by minimizing

$$arg\min_{\mathbf{V}} E_Q + wE_B, \tag{3}$$

where ω is the weighting factor that determines the degree of shrinkage in each iteration ($\omega = 0.3$ in most of our experimental results). To minimize the objective function, we formulate (3) using a least squares system in the form $\mathbf{Ax} = \mathbf{b}$:

$$\begin{bmatrix} Q \\ \overline{B} \end{bmatrix} [\mathbf{V}^{t+1}] = \begin{bmatrix} \left(p_{ij}^t\right)^2 \left(\mathbf{v}_i^t - \mathbf{v}_j^t\right) \\ \overline{\Omega \partial V^t} \end{bmatrix}, \tag{4}$$

where

$$\begin{aligned} Q_{ij} &= \begin{cases} 1 \ , & if\ j = e_{i1}, \\ -1, & if\ j = e_{i2}, \\ 0, & otherwise, \end{cases} \\ B_{ij} &= \begin{cases} w, & if\ j = r_i \in R, \\ 0, & otherwise, \end{cases} \\ \Omega_{ij} &= \begin{cases} w, & if\ i = j, \\ 0, & otherwise. \end{cases} \end{aligned} \tag{5}$$

R is the set of indices of the boundary voxels, and e_{i1} and e_{i2} denotes the two voxels of edge e_i, respectively. We solve the shrunk model $\mathbf{G}'$ by updating the voxel

TABLE 1
Parameters, Model Information, and Timing Statistics (Seconds)

	ω	α	Voxel number	Factorization	Iteration number	Back substitution	Thinning	Total
Candleholder	0.4	0.01	15571	0.861000	6	0.062000	1.767000	3.000000
Dino	0.3	0.01	22195	0.859000	6	0.072833	1.312000	2.607998
Horse	0.3	0.01	52058	4.265000	7	0.234429	1.579000	7.485003
Inner ear	0.3	0.01	42927	2.563000	6	0.153667	1.516000	5.001002
Portien	0.4	0.01	45455	3.328000	6	0.169333	4.031000	8.374998
Rockerarm	0.3	0.01	51508	3.766000	6	0.224000	1.203000	6.313000
Tap	0.3	0.01	36718	2.219000	8	0.152375	1.546000	4.984000
Torus Knot	0.3	0.01	77927	4.861000	5	0.384200	4.073000	10.85500
Triceratops	0.3	0.01	50565	4.475000	7	0.250000	4.235000	10.73500
Airplane	0.3	0.01	10990	0.406000	5	0.027800	0.750000	1.295000
Biplane	0.3	0.01	22218	1.371000	6	0.107333	5.337000	7.351998
Bishop	0.3	0.01	71659	6.515000	7	0.364000	2.672000	11.73500
Brontosaur	0.3	0.01	36647	2.328000	7	0.144714	1.094000	4.434998
Helicopter	0.3	0.01	14140	0.687000	7	0.046857	1.421000	2.435999
Rabbit	0.3	0.01	82154	9.891001	6	0.453167	2.578000	15.18800

positions $\mathbf{V}^t$ and the contraction ratio p_{ij}^t iteratively. Note that the model's (i.e., $\mathbf{G}$) structure remains unchanged during minimization. Therefore, it is not necessary to factorize the constant matrix $\mathbf{A}$ in each iteration. Applying backsubstitution to iteratively update the voxel positions is very efficient. We demonstrate the timing statistics in Table 1.

For the implementation detail, the shrinking process starts by precomputing the Cholesky factorization of the matrix $\mathbf{A}^T\mathbf{A}$ and then applying a back substitution to solve $\mathbf{V}^1$ (i.e., $\mathbf{V}^0 = \mathbf{V}$ is the voxel set of the original model). In the beginning, we consider $p_{ij}^0 = 0$; our system computes the new voxel set $\mathbf{V}^1$ according to this given information. Further computations are performed by recomputing the contraction ratios p_{ij}^t and using them to solve $\mathbf{V}^{t+1}$. This iterative solver shrinks the model to achieve zero volume. We terminate the iteration when the model is thin enough.

3.2.4 Stop Conditions

An iterative solver always needs a stop condition to terminate the repeated process. Our shrinking method stops when the shrinkage is close to zero. Therefore, we define the degree of shrinkage by computing the edge lengths before and after each step:

$$S(\mathbf{V}^t) = \frac{\sum_{\{i,j\}\in\mathbf{E}} \left|\mathbf{v}_i^t - \mathbf{v}_j^t\right|}{\sum_{\{i,j\}\in\mathbf{E}} \left|\mathbf{v}_i^0 - \mathbf{v}_j^0\right|} \quad \mathbf{v} \in \mathbf{V}. \tag{6}$$

Obviously, the degree of shrinkage $S(\mathbf{V}^t)$ decays when we shrink the model. Furthermore, the first derivative of $S(\mathbf{V}^t)$ also decays at the same time, and the value close to 0 suggests that the model is thin enough and is difficult to further shrink. We stop the shrinking process in this situation. In this paper, the process is terminated when $\partial S(\mathbf{V}^t)$ is lower than α ($\alpha = 0.01$ in all of our experimental results).

3.3 Iterative Thinning

We apply [32] when simplifying the shrunk model to a 1D skeleton. This method repeatedly removes boundary voxels by using its 26-adjacent connectivity. We record this connectivity before applying the shrinking algorithm. Since only the voxel positions are moved during the shrinking process, the connectivity of $\mathbf{G}$ and $\mathbf{G}'$ are the same. By applying the thinning algorithm on the two models, we can obtain skeletons with the same structures; only their appearances are different. Although applying the thinning algorithm to simplify the original model would result in jagged and unsmooth skeleton, applying the same method to our shrunk model would not have this problem. This is because the voxels are transferred to better positions. As for preserving the model's topology, Palágyi and Kuba have developed 14 masks and 12 directions to test if the boundary voxel can be removed or not. Our iterative thinning also enjoys this advantage due to their contributions. In this paper, the voxels remaining after the thinning algorithm are considered skeleton nodes.

The positions of the skeleton nodes change during the shrinking algorithm. Thus, we must connect nodes via their edges. Otherwise, they are only meaningless points (Fig. 4a). However, we cannot simply assume that the two skeleton nodes are connected if there exists a 26-adjacency between them; otherwise, the incorrect tiny loop with three edges would be potentially created after the edge connection (Fig. 4b). This mistake results from the ambiguous relationship of the thinned skeleton. For instance, in Fig. 5, the connectivity represented by red dotted lines produces a tiny three-edge loop, and the edges should not be all connected, although the nodes are adjacent. In this paper, we remove the loops with only three edges to overcome this problem.[1] We test if this connection causes a 3-edge cycle before it is added to the skeleton. The test is quite simple

1. It is possible to remove some very small and important loops. We can choose a smaller voxel size to avoid this problem. In this manner, any legal loop is represented in more detail with more edges and thus would not be deleted.

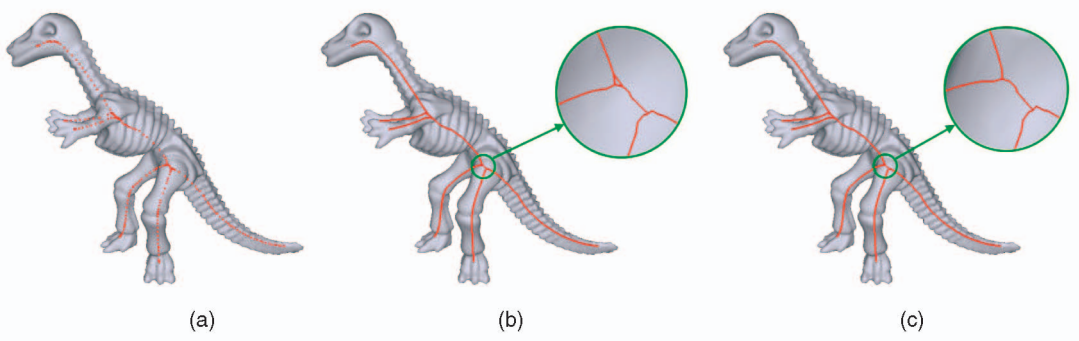

Fig. 4. In this figure, we show skeleton nodes and their connectivity. (a) The skeleton nodes are meaningless because there is no connectivity. (b) A skeleton with incorrect topology due to a naïve connection. (c) Connectivity repaired through our simple test.

since only the adjacent voxels require consideration. We show the results after connecting the skeleton nodes in Fig. 4c.

3.4 Branch Pruning

Many algorithms use a pruning step to remove unnecessary branches [2], [15]. Our method as well needs this step because the thinning algorithm leaves behind many unnecessary branches while extracting skeletons. As shown in Fig. 7a, although the skeleton's appearance appears to be free of redundant branches, the tiny branches still exist. This is because the shrinking method shortens the connected edges and, therefore, those unnecessary branches become much smaller. In our algorithm, the edges are not equally shortened since the boundary restriction E_b, and the various contraction ratios p_{ij} are used at different edges. Generally, from our observation, the branches with little shrinkage represent the model's features and thus are considered important. In other words, we prune the branches that are much more shortened after the shrinking process.

The connectivity of $\mathbf{G}$ and $\mathbf{G}'$ is the same since the shrinking algorithm only moves the voxels to new positions. One-to-one mapping between $\mathbf{G}$ and $\mathbf{G}'$ still exists. Thus, we can record the original position of each skeleton node and determine the amount of shrinkage by comparing the edge lengths before and after the shrinking algorithm. We measure the amount of shrinkage for the entire skeleton, formulated as $C_R = L'/L$, where L and L' denote the entire skeleton length before and after the shrinking algorithm, respectively. Similarly, we compute the shrinking ratio for each branch with the same method, denoted as C_P. The branches with larger amounts of shrinkage are then pruned because they are less important. Depending on our definition, the branch with lower C_P is shrunk more, and the branch with higher C_P maintains its length. Before pruning any branch, we select the smallest C_P and denote the value as C_S. We then discard the branch if its C_P is smaller than $(C_R + C_S)/2$. As shown in Fig. 6, the structure of a branch is sometimes changed if a branch is discarded. Therefore, after pruning each trivial branch, we need to refine all branch structures. This pruning algorithm stops when the smallest C_P is larger than $(C_R + C_S)/2$. Note that we do not execute the process if there are no redundant branches to be pruned. In the case of $C_R < 2C_S$, we skip the pruning step and do not remove any branches. Furthermore, the value of C_S does not change while branches are being discarded; our algorithm computes C_S only in the beginning and fixes it as a constant value. Otherwise, the value will repeatedly increase and important branches may be pruned.

In our experiment, our automatic pruning works well on many models. However, sometimes it is hard to determine if a branch is necessary or not because some of them are meaningful to humans. Hence, in addition to this automatic approach, we also let the user prune the branches using a tuning parameter. Let us denote C_U be the given threshold. Our system prunes the branch if its shrinking ratio C_P is smaller than C_U.

4 Results and Discussion

4.1 Experimental Evaluation

We apply our algorithm to extract curve skeletons from several polyhedral models and show the results in Fig. 9. In these experimental results, the redundant branches are automatically pruned without the input parameter C_U. To

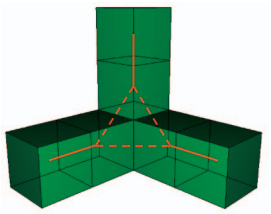

Fig. 5. In this figure, there are six skeleton nodes with their 26-adjacent connectivity represented by red lines. An internal loop is formed if all the red dotted lines are connected (i.e., the original topology is changed).

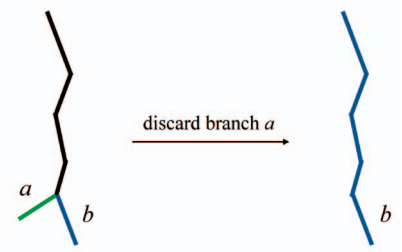

Fig. 6. Branches a and b are displayed in green and blue, respectively. Note that the structure of branch b is changed after branch a is discarded.

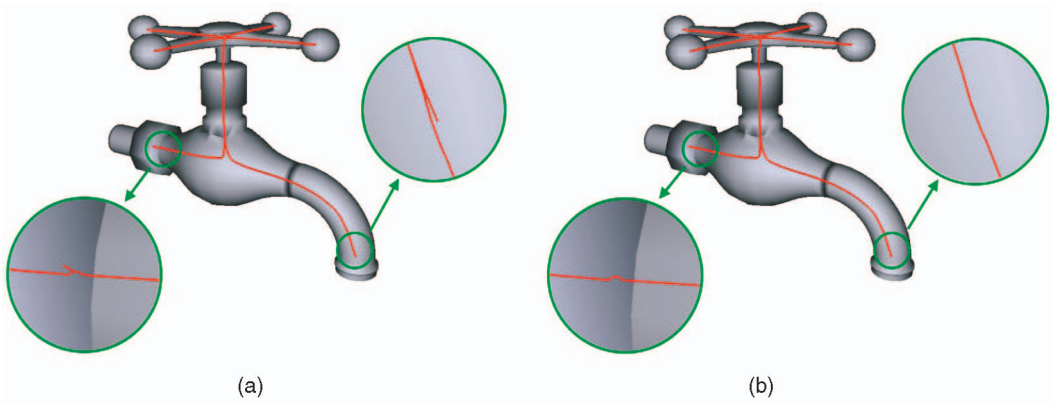

Fig. 7. The skeleton looks clean since the redundant branches are much shorter than before. However, we still have to prune the unnecessary branches to obtain a good structure. (a) The tiny branches on the tap. (b) The redundant branches are pruned by our algorithm.

test the robustness of our method, we also add 20 percent noise to each model and compare the results with the skeletons extracted from the original model. We compute the Peak Signal to Noise Ratio (PSNR) to determine the similarity of the skeletons of the model and the corresponding model with noise. For example, in the triceratops model, the PSNR value of its two skeletons is 64.62. In our experiments, the value of PSNR is between 50 and 60 since the skeletons are extracted from different models. In addition to noise, we also apply our algorithm to different poses of the same model. For example, in Fig. 10, we determine the curve skeletons of the kicking dinosaur and then obtain similar skeletons. This property will be useful in 3D model retrieval applications and will potentially allow for the retrieval of similar models with different poses based on the skeletons extracted using our algorithm. Our algorithm can also deal with the thin plate and degenerated models. Results are shown in Fig. 8.

We have implemented our algorithm in C and have run it on a Pentium 4 3.4-GHz PC. We solve the linear system using the Choleskey solver provided by the Taucs library [38]. Since factorizing the $\mathbf{A}^T\mathbf{A}$ matrix can be precomputed, each shrinking iteration needs only a back substitution. Thus, the total runtime in each of our experimental result takes only a few seconds. The factorization step takes the most computation time in our algorithm, which depends on the number of voxels and connected edges. In our experiment, it takes nearly 50 percent of the total runtime. We show the model information, as well as their detailed timing statistics, in Table 1.

4.2 Parameters

There are two major parameters required in our algorithm, including 1) the weighting factor ω used in (3), and 2) the stop condition α that terminates the shrinking process. Both parameters vary between 1.0 and 0.0. In general, the larger ω enhances the system when shrinking the model in the correct direction and more closely preserves the skeleton to the model's center. However, in this manner, the speed of shrinkage becomes slower with the completion of each step, and the iterative solver requires more iterations to shrink the object. The stop condition α determines how thin the shrunk model can be; the thinner model enhances the smoothness of the extracted skeleton. Similar to ω, the lower α demonstrates the better results but increases computations. Fig. 11 shows the influence of different ω and α when obtaining results. Obviously, the extracted skeletons are similar, although different parameters are given. In our experiment, we found that $\omega = 0.3$ and $\alpha = 0.01$ can adequately balance the quality, as well as the computation cost. We use these two values in most of our experimental results.

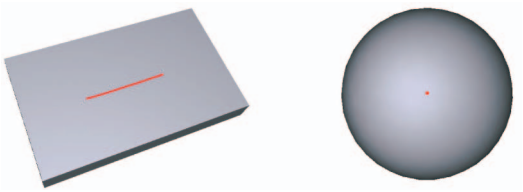

Fig. 8. The skeletons extracted from the thin plate and sphere models.

4.3 Properties

The skeletons extracted using our algorithm are thin and connected. Our algorithm enjoys the same advantages as the thinning method, since we apply the method to remove boundary voxels. In addition, the skeletons are smooth, although we still apply the thinning algorithm. This is because the energy function E_Q shortens the edge lengths and straightens each part of the skeleton. Therefore, the skeleton extends without unnecessary turns and becomes smoother. In this paper, we consider a curve smooth if the curvature of each interval is similar. To evaluate the smoothness of our skeleton, a measure function is proposed. We accumulate the magnitude of curvature variations between each pair of adjacent nodes, and use the value to indicate if the curve is smooth or not. In addition, we normalize this value so as to make the measurement independent of the model size. Obviously, the smaller value means smaller variations and, therefore, a smoother curve. To implement this idea, we introduce the following formula:

$$\begin{aligned} S_{moothness} &= \frac{1}{L}\sum_{\{i,j\}\in\mathbf{H}} |k_i - k_j|, \\ where \quad k_i &= \sum_{j\in M(i)} \frac{n_i - n_j}{|n_i - n_j|}, \quad n \in N, \end{aligned} \tag{7}$$

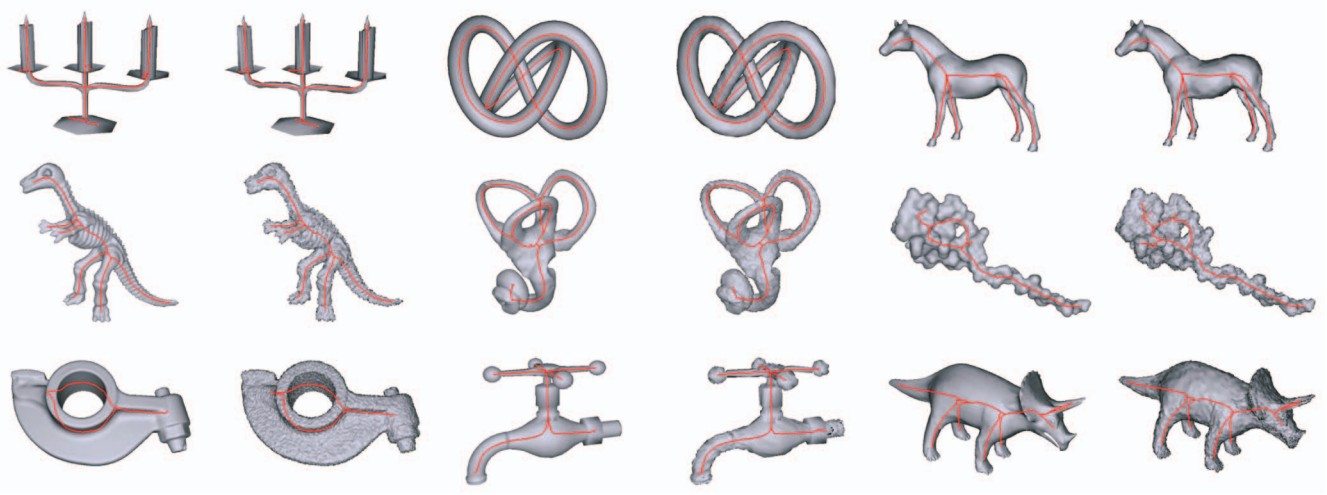

Fig. 9. We compare the skeletons determined from the model with and without noise. The extracted skeletons of the original models are shown in the odd columns from left to right. Similarly, the skeletons of their corresponding models with 20 percent noise are shown in the even columns. Obviously, our algorithm is less sensitive to noise since many skeletons of the original models are similar to those of noisy models.

L is the skeleton length, $\mathbf{N}$ and $\mathbf{H}$ denote the nodes and the edges of the skeleton, respectively, and $\mathbf{M}(i)$ represents the neighboring nodes of n_i. For the tap results shown in Figs. 2c and 2d, the smoothness of the skeleton extracted from the thinning method and our algorithm are 69.465 and 33.515, respectively. Our skeleton is smoother.

Depending on the definition, the skeleton extracted using our method does not exactly lie at the model's center, especially when a tiny ω is given. However, in order to satisfy boundary constraints while shortening edge lengths, the voxels must be moved toward the model's center; otherwise, the summation error of square lengths will rapidly increase. In other words, our two energy functions preserve the centering of the skeleton.

4.4 Comparison

We have compared our algorithm with an open software implemented in [14]. The algorithm extracts pretty good skeletons using only one parameter. However, the results highly depend on the input parameter, even when the value is only slightly changed. It requires the user to carefully choose the parameter, especially when some branches should be preserved, but some should be pruned. Fig. 12 shows the results determined from different parameters. It is clear that there are many redundant branches if an unsuitable value is chosen. In contrast to our proposed method, the two parameters are easily determined (Fig. 11). The larger ω, combined with smaller α, always achieves better results. In addition, the extracted skeletons are not sensitive to the parameter. The results are quite similar although we give different values for ω and α. As for the efficiency of the two methods, the method in [14] takes 86 and 583 seconds to compute the skeletons for the protein and triceratops models containing 12,381 and 17,496 vertices. Although the complexity of the volumetric data (i.e., number of voxels) is much higher, our algorithm is faster. It only takes 8 and 15 seconds to extract skeletons from the models containing 45,455 and 50,565 voxels. Both algorithms have been tested on a Pentium 4 3.4-GHz PC.

We also compare our algorithm to other methods, including the thinning [32], distance field [17], geometric [1], and general field [13] classes. However, it is not fair to extract skeletons by ourselves, since these algorithms use their own parameters to tune the results. We also have to make sure that the implementations are exactly the same. Fortunately, Cornea and Min [11] concludes the properties and applications of several existing methods. Therefore, we attempt to find the models shown in [11] and extract their curve skeletons using our algorithm. We also add noise to these models, since we claim that our algorithm is robust against surface perturbation. The models and their curve skeletons are shown in Fig. 13. Obviously, our algorithm extracts clean and similar skeletons whether noise is added or not. According to [11], only methods

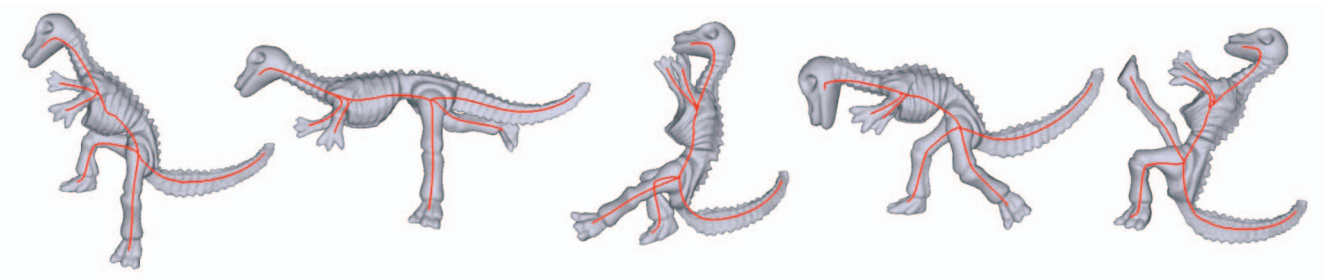

Fig. 10. Although the poses of the dinosaurs are different, the structures of the extracted skeletons are still the same.

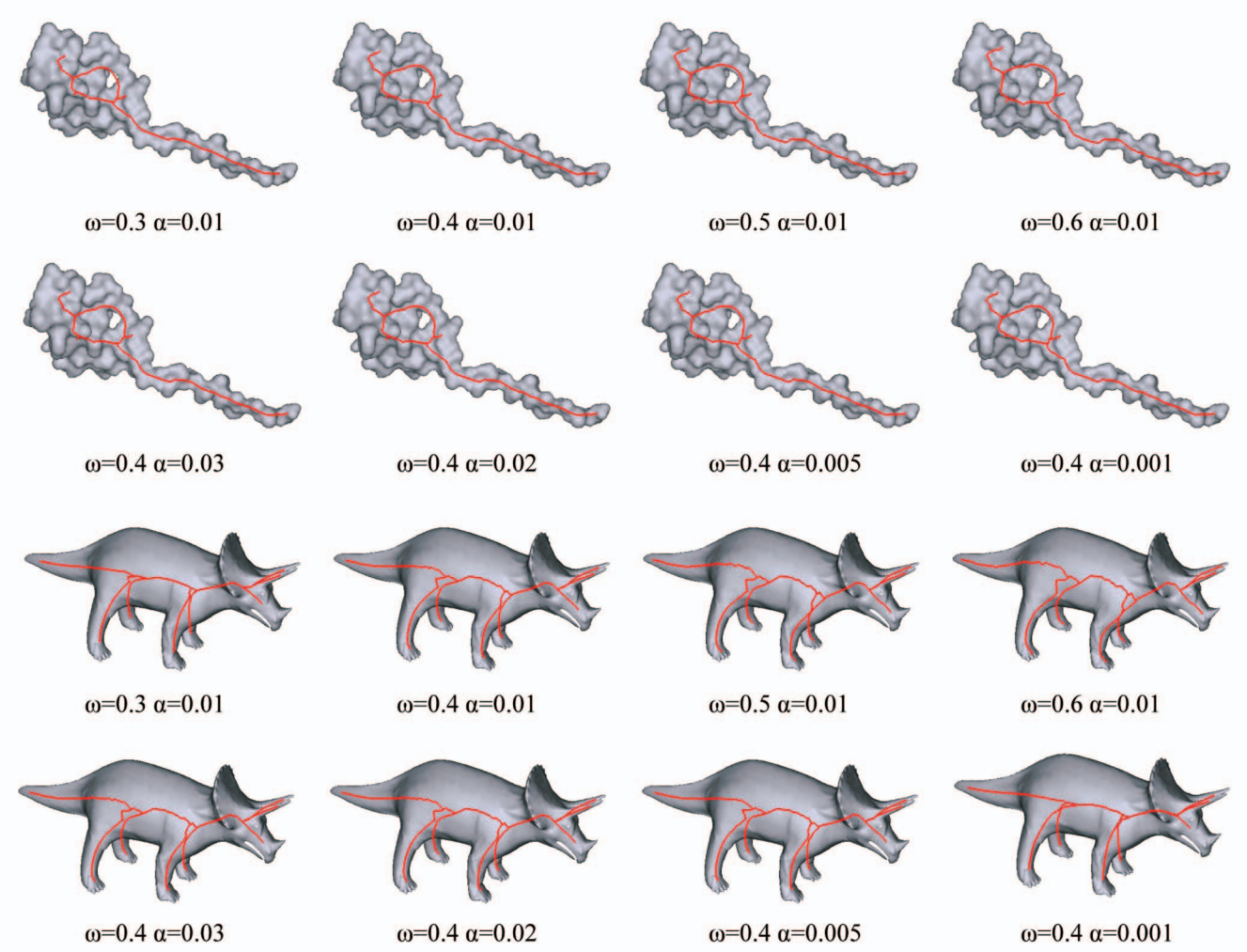

Fig. 11. This figure shows the results with different ω and α. Obviously, the extracted skeletons are similar in appearance. The results are insensitive to input parameters.

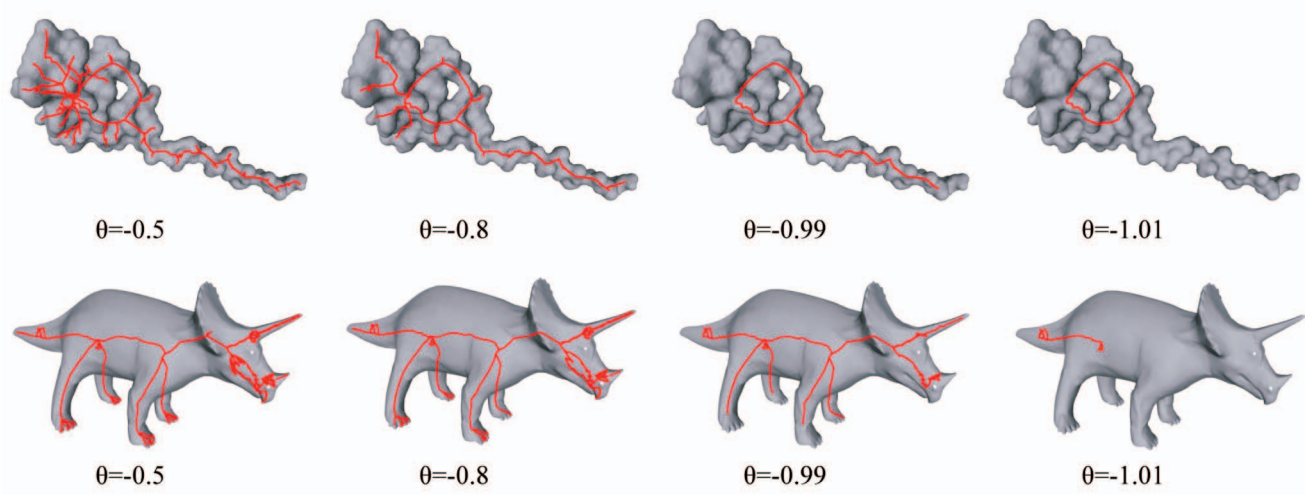

Fig. 12. We compare our algorithm with the algorithm presented in [14]. Although both methods extract good skeletons, [14] is sensitive to the input parameter. In contrast to that in Fig. 11, our algorithm is much more stable since the different parameters have far less influence on the results.

computing potential fields can extract clean and smooth skeletons. However, the computational complexity of these methods are $O(n^2)$, where n is the number of voxels that need more time to extract skeletons from larger models. In addition, they miss the brontosaur's tail and have many branches located at the rabbit's ear. As for the thinning method, although the algorithm is very efficient, the extracted skeletons are not smooth enough, since the algorithm is sensitive to noise. In addition, the skeleton found by the distance field method contains

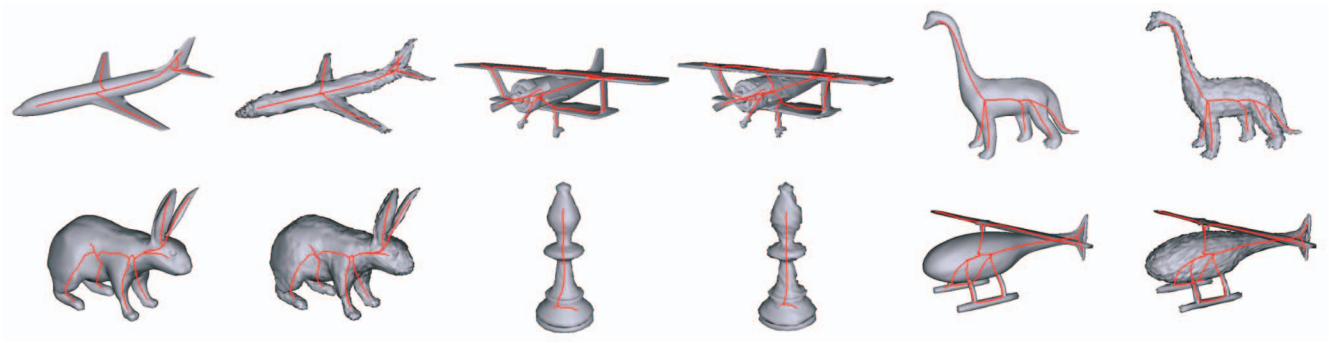

Fig. 13. We apply our algorithm to the models shown in [11] so as to compare the results with different methods.

many unnecessary branches, and many of them go out of the model entirely. The geometric method computes unusual results because the Reeb graph-based algorithm extracts the skeleton in a bottom-up direction. They need a nondirectional method to determine level sets. In summary, our method can extract clean and smooth skeletons with a lower computational cost while others cannot.

4.5 Limitations

Our algorithm extracts curve skeletons that are insensitive to perturbations on the model's surface. For example, in Fig. 14, the back and tail of the dinosaur have many jagged edges, but most of them are dismissed after our shrinking algorithm is applied. However, the algorithm still has space for improvement. Because we shorten the edges to thin the model, the resulting skeleton is straighter. Our method potentially causes the skeleton to diverge from the center of the model, especially thin positions, which tend to bend a lot. For example, in Fig. 15, we use a tiny ω in (3); the shrunk model is straight, and thus, the skeleton protrudes out of the candleholder. In order to reduce this problem, we can directly enlarge the value of ω.

Extracting skeletons from volumetric models requires the voxel size to be smaller than the features. Otherwise, they cannot correctly represent the model's details. In this situation, we need to use a smaller voxel size to represent features. However, this will significantly increase the model's complexity. Fig. 16 shows the hand skeletons extracted using different voxel sizes. Obviously, the voxel size should be very small since the fingers are close together. Otherwise, they would be considered a single component, and the wrong skeleton would be constructed. An intuitive way for solving this problem is to use an adaptive method that changes voxel sizes accordingly. We will include this improvement in our methods in the near future.

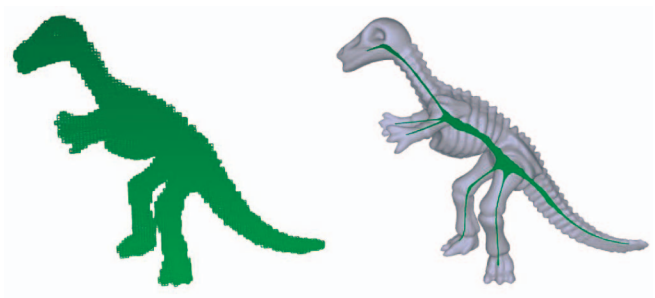

Fig. 14. The jagged edges on the dinosaur's back and tail are dismissed after the shrinking algorithm is applied.

5 Conclusion and Future Work

In conclusion, we have introduced a new and novel approach for automatically extracting skeletons. It is based on iterative least squares optimization that shrinks models and applies the thinning algorithm to extract 1D skeletons. Our algorithm takes less computation than other methods that use the general field function and is more robust than methods based on the distance field and thinning algorithms. Several improvements to our algorithm will be completed in the near future. We will attempt to prevent the skeleton from diverging from the model's center. In addition, the adaptive volumetric models are also useful in extracting skeletons because many regions of the model can be represented using larger voxels. Therefore, the number of voxels and edges are effectively decreased and computations, in many aspects, will be reduced.

Acknowledgments

The authors would like to thank Dey and Sun [14] for providing their open software and for helping with the experimental comparisons between their methods and the methods presented in this paper. The authors would also like to thank the anonymous reviewers for helping in the improvement of this paper. This work is supported by the Landmark Program of the NCKU Top University Project under Contract B0008 and is supported in part by the National Science Council under Contracts NSC-95-2221-E-006-193-MY2 and NSC-96-2628-E-006-200-MY3.

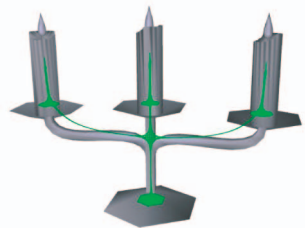

Fig. 15. The crutch is diverged from the center of the candleholder.

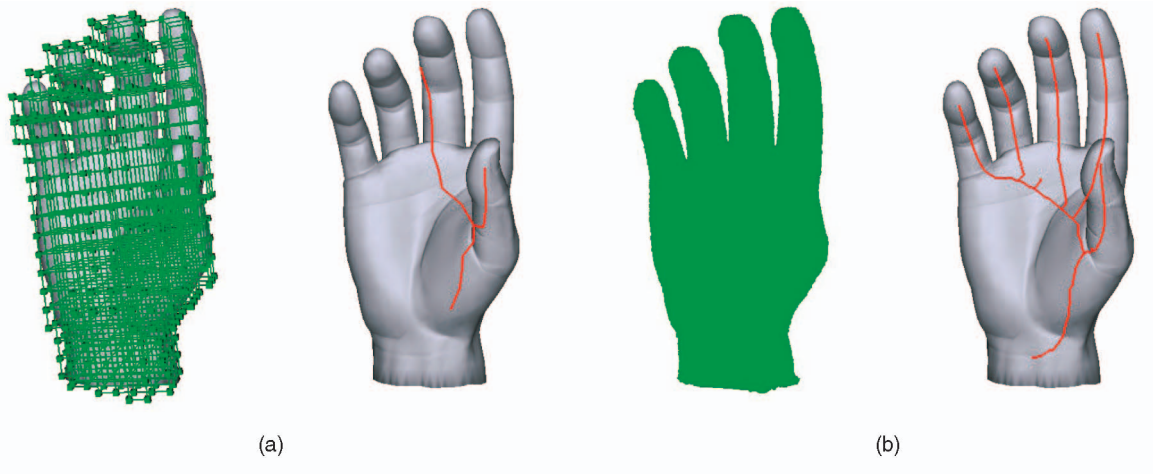

Fig. 16. In this figure, there are two volumetric models with different voxel sizes. (a) The distances between the fingers are smaller than the voxel size; the represented fingers are all connected together, and thus, only one branch is extracted from the four fingers. (b) The voxel size is small enough to separate the fingers; thus, the finger branches can be extracted. (a) Voxel size $= 0.05$. (b) Voxel size $= 0.005$.

REFERENCES

[1] N. Amenta, S. Choi, and R.K. Kolluri, "The Power Crust," *Proc. Sixth ACM Symp. Solid Modeling and Applications (SMA '01)*, pp. 249-266, 2001.

[2] G. Aujay, F. Hetroy, F. Lazarus, and C. Depraz, "Harmonic Skeleton for Realistic Character Animation," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA '07)*, pp. 151-160, 2007.

[3] S.R. Aylward and E. Bullitt, "Initialization, Noise, Singularities and Scale in Height Ridge Traversal for Tubular Object Centerline Extraction," *IEEE Trans. Medical Imaging*, vol. 21, no. 2, pp. 61-75, 2002.

[4] S.R. Aylward, J. Jomier, S. Weeks, and E. Bullitt, "Registration and Analysis of Vascular Images," *Int'l J. Computer Vision*, vol. 55, no. 2-3, pp. 123-138, 2003.

[5] G. Bertrand and Z. Aktouf, "A 3D Thinning Algorithm Using Subfields," *Vision Geometry*. SPIE, pp. 113-124, 1994.

[6] J. Bloomenthal, "Medial-Based Vertex Deformation," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA '02)*, pp. 147-151, 2002.

[7] S. Bouix and K. Siddiqi, "Divergence-Based Medial Surfaces," *Proc. Sixth European Conf. Computer Vision (ECCV '00)*, pp. 603-618, 2000.

[8] J.W. Brandt and V.R. Algazi, "Continuous Skeleton Computation by Voronoi Diagram," *CVGIP: Image Understanding*, vol. 55, no. 3, pp. 329-338, 1992.

[9] A. Brennecke and T. Isenberg, "3D Shape Matching Using Skeleton Graphs," *Proc. Simulation and Visualization Conf. (SimVis '04)*, pp. 299-310, 2004.

[10] J.-H. Chuang, C.-H. Tsai, and M.-C. Ko, "Skeletonization of Three-Dimensional Object Using Generalized Potential Field," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1241-1251, Nov. 2000.

[11] N.D. Cornea and P. Min, "Curve-Skeleton Properties, Applications, and Algorithms," *IEEE Trans. Visualization and Computer Graphics*, vol. 13, no. 3, pp. 530-548, May/June 2007.

[12] N.D. Cornea, D. Silver, and P. Min, "Curve-Skeleton Applications," *Proc. IEEE Visualization Conf. (VIS '05)*, p. 13, 2005.

[13] N.D. Cornea, D. Silver, X. Yuan, and R. Balasubramanian, "Computing Hierarchical Curve-Skeletons of 3D Objects," *The Visual Computer*, vol. 21, no. 11, pp. 945-955, 2005.

[14] T.K. Dey and J. Sun, "Defining and Computing Curve-Skeletons with Medial Geodesic Function," *Proc. Symp. Geometry Processing*, pp. 143-152, 2006.

[15] H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological Persistence and Simplification," *Proc. 41st Ann. Symp. Foundations of Computer Science (FOCS '00)*, p. 454, 2000.

[16] S. Fang and H. Chen, "Hardware Accelerated Voxelization," *Computers and Graphics*, vol. 24, no. 3, pp. 433-442, 2000.

[17] N. Gagvani and D. Silver, "Parameter-Controlled Volume Thinning," *CVGIP: Graphical Models and Image Processing*, vol. 61, no. 3, pp. 149-164, 1999.

[18] N. Gagvani and D. Silver, "Animating Volumetric Models," *Graphical Models*, vol. 63, no. 6, pp. 443-458, 2001.

[19] T. He, L. Hong, D. Chen, and Z. Liang, "Reliable Path for Virtual Endoscopy: Ensuring Complete Examination of Human Organs," *IEEE Trans. Visualization and Computer Graphics*, vol. 7, no. 4, pp. 333-342, Oct.-Dec. 2001.

[20] L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He, "Virtual Voyage: Interactive Navigation in the Human Colon," *Proc. SIGGRAPH '97*, pp. 27-34, 1997.

[21] A. Kanitsar, D. Fleischmann, R. Wegenkittl, P. Felkel, and M.E. GrLoller, "CPR: Curved Planar Reformation," *Proc. IEEE Visualization Conf. (VIS '02)*, pp. 37-44, 2002.

[22] A. Kanitsar, R. Wegenkittl, D. Fleischmann, and M.E. GrLoller, "Advanced Curved Planar Reformation: Flattening of Vascular Structures," *Proc. IEEE Visualization Conf. (VIS '03)*, pp. 43-50, Oct. 2003.

[23] S. Katz and A. Tal, "Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts," *Proc. SIGGRAPH '03*, pp. 954-961, 2003.

[24] F. Lazarus and A. Verroust, "Level Set Diagrams of Polyhedral Objects," *Proc. Fifth ACM Symp. Solid Modeling and Applications (SMA '99)*, pp. 130-140, 1999.

[25] T.-Y. Lee, C.-H. Lin, H.-K. Chu, Y.-S. Wang, S.-W. Yen, and C.-R. Tsai, "Mesh Pose-Editing Using Examples," *Computer Animation Virtual Worlds (COMPUTER ANIMATION and SOCIAL AGENTS '07)*, vol. 18, nos. 4-5, pp. 235-245, 2007.

[26] T.-Y. Lee, Y.-S. Wang, and T.-G. Chen, "Segmenting a Deforming Mesh into Near-Rigid Components," *The Visual Computer*, Proc. Pacific Conf. Computer Graphics and Applications (Pacific Graphics '06), vol. 22, no. 9, pp. 729-739, 2006.

[27] F.F. Leymarie, "Three-Dimensional Shape Representation via Shock Flows," PhD dissertation, Division of Eng., Brown Univ., May 2003.

[28] X. Li, T.W. Toon, and Z. Huang, "Decomposing Polygon Meshes for Interactive Applications," *Proc. Symp. Interactive 3D Graphics (I3D '01)*, pp. 35-42, 2001.

[29] C.-H. Lin and T.-Y. Lee, "Metamorphosis of 3D Polyhedral Models Using Progressive Connectivity Transformations," *IEEE Trans. Visualization and Computer Graphics*, vol. 11, no. 1, pp. 2-12, Jan./Feb. 2005.

[30] P.-C. Liu, F.-C. Wu, W.-C. Ma, R.-H. Liang, and M. Ouhyoung, "Automatic Animation Skeleton Construction Using Repulsive Force Field," *Proc. 11th Pacific Conf. Computer Graphics and Applications (PG '03)*, p. 409, 2003.

[31] C.M. Ma and M. Sonka, "A Fully Parallel 3D Thinning Algorithm and Its Applications," *Computer Vision and Image Understanding*, vol. 64, no. 3, pp. 420-433, 1996.

[32] K. Palágyi and A. Kuba, "A Parallel 3D 12-Subiteration Thinning Algorithm," *Graphical Models and Image Processing,* vol. 61, no. 4, pp. 199-221, 1999.
[33] S.M. Pizer, D.S. Fritsch, P.A. Yushkevich, V.E. Johnson, and E.L. Chaney, "Segmentation, Registration and Measurement of Shape Variation via Image Object Shape," *IEEE Trans. Medical Imaging,* vol. 18, no. 10, pp. 851-865, 1999.
[34] T.B. Sebastian, P.N. Klein, and B.B. Kimia, "Shock-Based Indexing into Large Shape Databases," *LNCS,* vol. 2352, pp. 731-746, 2002.
[35] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson, "Skeleton Based Shape Matching and Retrieval," *Proc. Shape Modeling Int'l (SMI '03),* p. 130, 2003.
[36] S. Takahashi, T. Ikeda, Y. Shinagawa, T.L. Kunii, and M. Ueda, "Algorithms for Extracting Correct Critical Points and Constructing Topological Graphs from Discrete Geographical Elevation Data," *Computer Graphics Forum,* vol. 14, no. 3, pp. 181-192, 1995.
[37] J. Tierny, J.-P. Vandeborre, and M. Daoudi, "3D Mesh Skeleton Extraction Using Topological and Geometrical Analyses," *Proc. 14th Pacific Conf. Computer Graphics and Applications (PG '06),* pp. 85-94, 2006.
[38] S. Toledo, D. Chen, and V. Rotkin, *TAUCS: A Library of Sparse Linear Solvers Version 2.2.* Tel-Aviv Univ., http://www.tau.ac.il/stoledo/taucs/, Sept. 2003.
[39] L. Wade and R.E. Parent, "Automated Generation of Control Skeletons for Use in Animation," *The Visual Computer,* vol. 18, no. 2, pp. 97-110, 2002.
[40] M. Wan, F. Dachille, and A. Kaufman, "Distance-Field Based Skeletons for Virtual Navigation," *Proc. IEEE Visualization Conf. (VIS '01),* pp. 239-246, 2001.
[41] F.-C. Wu, W.-C. Ma, R.-H. Liang, B.-Y. Chen, and M. Ouhyoung, "Domain Connected Graph: The Skeleton of a Closed 3D Shape for Animation," *The Visual Computer,* vol. 22, no. 2, pp. 117-135, 2006.

Yu-Shuen Wang received the BS degree from the National Cheng Kung University, Tainan, Taiwan, R.O.C., in 2004. He is currently a PhD candidate in the Department of Computer Science and Information Engineering, National Cheng Kung University. His research interests include computer graphics, mesh segmentation, skeletonization, and computer animation.

Tong-Yee Lee received the PhD degree in computer engineering from Washington State University, Pullman, in May 1995. He is currently a professor in the Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan. His current research interests include computer graphics, nonphotorealistic rendering, image-based rendering, visualization, virtual reality, surgical simulation, medical visualization and medical system, and distributed and collaborative virtual environments. He is an associate editor for the *IEEE Transactions on Information Technology in Biomedicine* from 2007 to 2010. He is also on the editorial advisory board of the *Journal Recent Patents on Engineering*, an editor of the *Journal of Information Science and Engineering*, and a region editor of the *Journal of Software Engineering*. He served as a member of the international program committees of several conferences including IEEE Visualization, Pacific Graphics, the IEEE Pacific Visualization Symposium, the IEEE-EMBS International Conference on Information Technology and Applications in Biomedicine, the International Conference on Artificial Reality and Telexistence, and the International Conference in Central Europe on Computer Graphics, Visualization, and Computer Vision. He leads the Computer Graphics Group, Visual System Laboratory, National Cheng-Kung University (http://graphics.csie.ncku.edu.tw/). He is a member of the IEEE and the ACM.