Geodesic-based Skeleton Smoothing

Porawat Visutsak and Korakot Prachumrak

***Abstract*—** Skeleton is at the main interest of 3D character animation. The most common techniques for skeleton computing are based on the Reeb graph and the shortest path finding. Using only the shortest path algorithms for extracting the critical points and constructing the Reeb graph over the surface of the model may generate unwanted skeleton joints. In this paper, we present a new approach to compute the skeleton of the 3D meshed model in a Riemannian space, based on Blum's Medial Axis Transform and geodesic distance algorithm. We gain the benefit of geodesic distance functions and parameterization that allow for efficient handling of topological changes of dynamics curves and surfaces. Thus, our approach can provide the robustness against any changes of a rotation and/or a translation of the 3D mesh model. We are able to generate one-voxel thick, graph-like skeleton. Han and Poston's Chord-to-point Distance Accumulation then be applied for adjusting the locations of consecutive points along the skeleton. The smoothed skeleton is split in order to create segments and joints corresponding to its shape. The new skeleton can be regenerated later on. Therefore, the new skeleton produced from our method can capture the essential shape characteristics in a compact form, while preserving the meaningful anatomical information of the 3D character models. The demonstration of the approach with several examples is also provided.

***Keywords*—** Skeleton; Skeleton smoothing; Geodesic distance; Medial axis transform; Riemannian space; Chord-to-point distance accumulation.

I. Introduction

THE skeleton is a basic structure of representing the 3D objects, frequently used in computer graphics, especially in the areas of character animation and 3D modeling. Using a skeleton as an abstraction of an object has two major benefits. First, it can contain both shape features and topological structures of an original object. Another benefit depends on its characteristic to capture the essential shape of a 3D object in a low-dimension form. Numerous algorithms have been developed to generate the skeleton in graph-like or a curve-like form [7], [8], [10], [11], [12]. Unfortunately, these approaches do not suit for producing a skeleton for use in animation since they need the additional processes to eliminate the redundant skeleton branches that may generate during the skeleton extraction process. The most common technique to represent the 3D objects, that has been the standard for many years, is the Reeb graph [11], originally defined by Reeb. The Reeb graph is obtained by applying the continuous function, usually a height function, to encode the topological structure. Using a height function to build a Reeb graph does not guarantee that the graph is invariant to the affine transformations, which is an essential feature of the skeletal structure of the model [12]. The extended version of the Reeb graph, called the Multi-Resolution Reeb graph (MRG) [7] has been proposed. To construct the MRG, the topological characteristics of the shape must be defined in terms of the critical points of a function on the manifold; this function is called a mapping function. The mapping function maps the points from the manifold of the shape to the domain of the function, and the configuration of the critical points of the mapping function can represent the topology of the shape [14]. This configuration can be embedded by the Reeb graph, and becomes an essential property of the shapes later on. When the mapping function is defined, the model is then partitioned into regions that correspond to equal intervals of the mapping function [15]. Each partition of the model is represented as a node in the Reeb graph, and adjacent nodes are linked by an edge that connects the corresponding nodes.

To apply skeleton for use in character animation and 3D modeling, skeleton animation is a common technique for animating a 3D model. Controlling the movement of a skeleton in a way that is designed to appear naturally is accomplished using a control skeleton (sometimes called an Inverse Kinematics or IK skeleton). IK skeleton is an articulated structure of segments and joints combined with information detailing how the surface geometry of the figure is anchored to that structure. Recent work on semi-automatic skeleton extraction is introduced by Aujay et al. [2]. This system allows users select the starting point on the character model, and then it generates a skeleton to match the ones that are created by hand by professionals in most biped and quadruped cases. A method for fully automatic generation of a control skeleton is proposed by Wade and Parent [18]. The main task of the system involves discretizing the figure, computing its discrete medial surface (DMS), and then using the discrete medial surface both to create the skeleton and to attach the vertices of the model to that structure. However, a major drawback of Wade and Parent's method is only features with a size greater than the voxel size can be taken into account. This often leads to computationally expensive algorithms.

In this paper, we present a novel approach to compute the skeleton of 3D mesh model based on Blum's Medial Axis Transform and geodesic distance algorithm. Blum et al. [4] propose to use the medial axis to define the skeleton. The algorithm for computing the skeleton in terms of the medial axis is often refers to as the "grassfire" algorithm. The main idea of the "grassfire" algorithm is lighting a fire, started from the border of the object, and let it burn into the object at a

constant speed; it will then meet in the medial axis. One starts on the border of the object and strips away one layer of pixels after another until one reaches points that fire reaches from two directions [1]. The medial axis transform of the region is the set of points reached by more than one fire at the same time. Unfortunately, the medial axis transform does not provide robustness against a rotation and a translation of the objects. Therefore, a more sophisticated algorithm is needed in order to solve this problem, and that algorithm is the geodesic distance functions. By using the geodesic approximation algorithm proposed by [9] to compute the "single source, all destination" shortest path on a surface of the model, any changes of a rotation and/or a translation do not affect the value of the function. Thus, this becomes the major property of the function of geodesic distance that gives the advantage of being invariant to a rotation and a translation. The basic idea of our approach is to iteratively snip off the spurious skeleton joints that have been produced from the step of the branch region determination. Applying the function of geodesic distance can guarantee that the new approach is invariant to a rotation and a translation, and it is also robust against the changes in the connectivity on the 3D shapes. Unfortunately, this raises the problem of producing meaningless joints since the location of skeleton joints does not match the real bone structure of the model. Thus, the additional operation is needed, by using the filtering process we can obtain the smoothed skeleton. Chord-to-point distance accumulation [6] then be applied in order to split the smoothed skeleton into the segments, and then the new joints are located correspond to chord-to-point distance accumulation values. A brief review of the medial axis transform and the geodesic distance function are illustrated in sections 2. The skeleton smoothing method based on Chord-to-point is outlined in section 3; we also demonstrate the usability of the smoothed 3D skeleton with several figures, and the experimental results are shown in section 4. Finally, the conclusion and the evaluation of our approach are also provided.

II. Geodesic-based Skeleton Smoothing

A. The Medial Axis Transform

The idea of using a skeleton as an abstraction of the shape goes back to [4]. Blum et al. define a skeleton in terms of the medial axis (MA), which is a set of curves that roughly run along the middle of an object, as shown in figure 1. According to [4], the medial axis of a curve S is the locus of the centers of the maximal disks contained in S. Let R^n be a symmetry set (where n is a number of dimensional space) which is defined similarly to the medial axis, except that it also includes the circles not contained in S and thus the medial axis is a subset of the symmetry set. The medial axis can then be defined as follows:

Definition 1. Let S be an arbitrary curved surface; Let $D^n(p, r)$ be a closed disk with a radius r centered at a point p, where $S(p, r) \subseteq R^n$. A maximal disk in S is a closed disk $D^n(p, r)$ contained in S.

Property 1. If X is a maximal disk in S, then S is not properly contained in any other closed disk in S.

Definition 2. Let $S(p, r) \subseteq R^n$. The medial axis (MA) of S is the locus of the centers of the maximal disks contained in S. The medial axis of a 3D object denoted R^3 is sometimes called the medial surface. The continuous function of a real-value that assigns to each center of a maximal disk in S is called the radius function of that medial axis.

The medial axis together with the associated radius function of the maximal disks is called the medial axis transform (MAT). Then, the medial axis can be defined:

Definition 3. The medial axis transform (MAT) of an object consists of its medial axis together with the associated radius function.

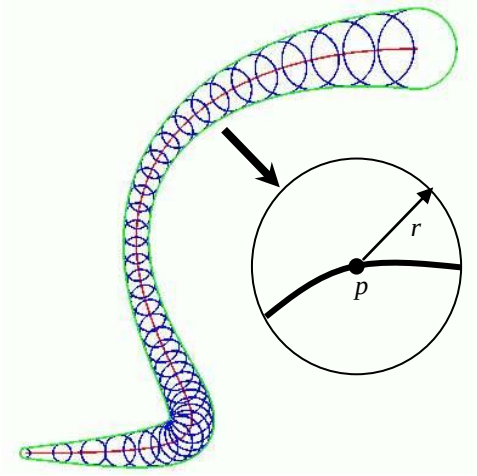

Figure 1. The medial axis of an object.

B. The Geodesic Distance Function

A Riemannian space is a mathematical geometry concept that studies curves and surfaces in higher dimensions, giving a precise meaning to concepts like angle, length, area, volume and curvature. On a Riemannian system, the geodesic distance is the distance between two points on the surface of the model, computed by using the Dijkstra's algorithm to find the shortest path between the two points on the surface made up by n points [14]. The following definitions characterize the geodesics based on [9].

Definition 4. Given R^3 be a surface in a Riemannian space G, and source vertex $v_s \in G$, an explicit representation of the geodesic distance function D: $G \rightarrow R^3$. For any point $p \in G$, this function $D(p)$ returns the length of the geodesic path from p back to the source v_s. The approximation of $D(p)$ can be given by

$$||x_i - x_j|| \approx D(x_i, x_j) \text{ when } x_j \approx x_j$$

Consider the length of a curve S, represented in R^3 by

equations

$$S: x^i = x^i(t), t_1 \le t \le t_2,$$

Definition 5. The distance *s* between two points t_1 and t_2 on a curve $x^\gamma = x^\gamma(t)$ in R^3 is given by

$$s = \int_{t_1}^{t_2} \sqrt{g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta}\, dt, \qquad (\infty, \beta = 1, \dots, n). \quad (1)$$

The minimum of (1) will be yielded a geodesic of the space, by using Euler's or Lagrange's equations:

$$\frac{\partial F}{\partial x^k} - \frac{d}{dt}\left(\frac{\partial F}{\partial \dot{x}^k}\right) = 0$$

with

$$F = \sqrt{g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta}$$

we have

$$\frac{\partial F}{\partial x^k} = \frac{1}{2}(g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta)^{-1/2}\, \frac{\partial g_{\infty\beta}}{\partial x^k}\dot{x}^\infty \dot{x}^\beta$$

$$\frac{\partial F}{\partial \dot{x}^k} = \frac{1}{2}(g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta)^{-1/2}\, 2g_{\infty k}\dot{x}^\infty$$

since

$$\frac{ds}{dt} = \sqrt{g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta}$$

Euler's equations can be written in the form

$$\frac{d}{dt}\left(\frac{g_{\infty k}\dot{x}^\infty}{\dot{s}}\right) - \frac{1}{2\dot{s}}\frac{\partial g_{\infty\beta}}{\partial x^k}\, \dot{x}^\infty \dot{x}^\beta = 0,$$

and, carrying out the indicated differentiation, we obtain

$$g_{\infty k}\ddot{x}^\infty + \frac{\partial g_{\infty k}}{\partial x^\beta}\, \dot{x}^\infty \dot{x}^\beta - \frac{1}{2}\frac{\partial g_{\infty\beta}}{\partial x^k}\, \dot{x}^\infty \dot{x}^\beta = \frac{g_{\infty k}\dot{x}^\infty \ddot{s}}{\dot{s}}$$

by writing

$$\frac{\partial g_{\infty k}}{\partial x^\beta}\, \dot{x}^\infty \dot{x}^\beta = \frac{1}{2}\left(\frac{\partial g_{\infty k}}{\partial x^\beta} + \frac{\partial g_{\beta k}}{\partial x^\infty}\right) \dot{x}^\infty \dot{x}^\beta$$

we obtain

$$g_{\infty k}\ddot{x}^\infty + [\infty\beta, k]\, \dot{x}^\infty \dot{x}^\beta = \frac{g_{\infty k}\dot{x}^\infty \ddot{s}}{\dot{s}}$$

if we use curve length as parameter, $\dot{s} = 1, \ddot{s} = 0$, then the equation becomes

$$g_{\infty k}\ddot{x}^\infty + [\infty\beta, k]\, \dot{x}^\infty \dot{x}^\beta = 0$$

multiplying by $g^{\gamma k}$, we obtain

$$\ddot{x}^\gamma + [\infty\beta, r]\, \dot{x}^\infty \dot{x}^\beta = 0 \quad (2)$$

These are the desired equations of geodesics. In equation (2), dots denote the differentiation with respect to the length parameter of the curve *S*. According to [3], the geodesic distance function given by Eq. (2) is rotation and translation invariant.

III. Implementation Details

A. Pruning the skeleton

According to [16], the following definitions are used to formally define the skeleton in our approach:

Definition 6. The *degree* of a point is defined as a finite number of points in its 26-neighborhood (figure 2 shows the 26-neighborhood structure).

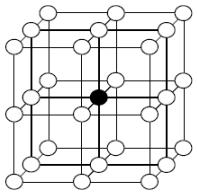

Figure 2. The 26-neighborhood structure.

Definition 7. By the assumption that the skeleton is one-voxel thick, the skeleton *end point* is a point that has a degree one. The *middle point* is a point that has a degree two. The *connection point* is a point that has a degree three or higher and all the other neighbors are either the end points or the middle points. The *branched connection point* is a point that has a degree three or higher and at least one of its neighbors is neither an end point nor a middle point. The 26-connected branched connection points form the *branched region*.

Algorithm 1 shows the steps of pruning the skeleton, the original 3D object, and the result of pruning are shown in figure 3. The method involves of nine different steps:

(1) Compute the degree of each point.
(2) Determine which point are the end points, the middle points, the connection points, and the branched connection points. If there are no branched connection points, exit the program.
(3) Organize the branched connection points into the branched regions. Each branched region consists of 26-adjacent branched connection points.
(4) In each branched region, find all the end points and the middle points that are 26-adjacent to this branched region, and

Input: A set of points on the surface of model, $P = \{p_1, p_2, \ldots, p_n\}$.
Output: The pruned skeleton of the given model.
// Read P and determine which p is an endpoint, a middle point, a connection point, and a branched connection point.
for all $p \in surface$
compute *degree of p*
if *degree of p* = 1
$p \leftarrow$ *endpoint*
if *degree of p* = 2
$p \leftarrow$ *middle point*
if *degree of p* $\geq$ 3 **and** all the other neighbors are either the end points or the middle points
$p \leftarrow$ *connection point*
if *degree of p* $\geq$ 3 **and** at least one of its neighbors is neither an end point nor a middle point
$p \leftarrow$ *branched connection point*
store *p* in *ListOfBranchedConnectionPoint*
end if
end if
end if
end if
end for
// Construct a branched region from p in ListOfBranchedConnectionPoint.
for all $p \in ListOfBranchedConnectionPoint$
if *NumberOfNeighbours*(*p*) = 26
construct *BranchedRegion*(*p*)
end if
end for
// Find the end points and the middle points that are 26-adjacent to this branched region, and mark each as "terminator".
for all $p \in BranchedRegion$
if *degree of p* = 1
$p \leftarrow$ *endpoint*
$p \leftarrow$ *terminator*
if *degree of p* = 2
$p \leftarrow$ *middle point*
$p \leftarrow$ *terminator*
end if
end if
end for
// Determine the centroids in each branched region.
// Compute the geodesic distance and find the shortest path correspond to each centroid.
for all $p \in BranchedRegion$
$CentroidOfBranchedRegion = (\sum x_i/N, \sum_{i=1}^{N} y_i/N, \sum_{i=1}^{N} z_i/N)$
if $p \approx CentroidOfBranchedRegion$
$p \leftarrow$ *centroid*
geod_dist (p, p_i) // compute the geodesic distance from centroid to all p_i, where p_i = terminator
Dijkstra's_shortest_path (p, p_i) // apply the *Dijkstra's* algorithm
// to find the shortest path from centroid to all p_i
construct *ShortestPath*(*P*) where $P = (p, p_i)$
end if
end for
// Remove the branched connection points that are not on any of the shortest paths**.**
// Construct the pruned skeleton.
for all $p \in BranchedRegion$
for all $p \in ListOfBranchedConnectionPoint$
if $p \notin ShortestPath$
remove *p*
update *ListOfBranchedConnectionPoint*
construct *PrunedSkeleton*(*p*) // construct the pruned skeleton from the remaining points in the list
end if
end for
end for

Algorithm 1 Skeleton pruning method.

mark each as "terminator".
(5) Determine the centroids, based on Blum's definition of a skeleton; the skeleton points must be the centers of the maximal disks contained in the curve *C*.
(6) Compute the geodesic distance from the particular centroids to all points in step 4.
(7) Apply the Dijkstra's algorithm, in order to find the shortest paths between the centroids and their "terminator".
(8) Remove the branched connection points that are not on any of the shortest paths.
(9) Generate the pruned skeleton from the remaining points.

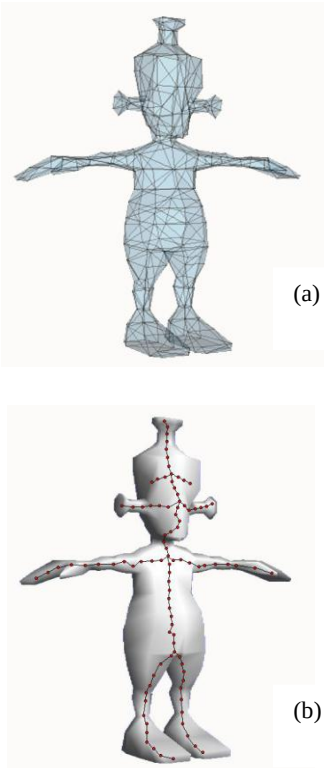

Figure 3. (a) The original 3D mesh model (b) Constructing the pruned skeleton.

B. Smoothing the skeleton

Once we can eliminate a large number of spurious skeleton joints, the next problem is raised which is the meaningless points that align unreasonably along a skeleton. These points can affect on the later creation of segments and joints, therefore the pruned skeleton is subjected to a smoothing operation. The smoothing process involves calculating the average position of consecutive points along the pruned skeleton, reassigning the new joints by splitting a skeleton into the segments [17].

i) Adjusting the skeleton

The pruned skeleton has been originated with a one-voxel thick, in a graph-like form as shown in figure 3 (b). Before we apply chord-to-point distance accumulation to the pruned skeleton, the skeleton needs to be smoothened as nearly as a curve. This can be accomplished by using the average filtering over a sliding window consisting of five consecutive points to adjust the positions of consecutive points along the skeleton.

Let $C = \{p_i = (x_i, y_i, z_i), i = 1,\ldots, N\}$ be the set of coordinate of points on the skeleton. The ith point in the set is denoted by p_i and p_{i+1} is its neighboring point. To calculate the average at the point p_i, we make an approximation of the change in the positions of two consecutive points before p_i and two consecutive points after p_i and divide it by 5 (the number of points between $p_{i-2}, \ldots, p_i, \ldots, p_{i+2}$). Thus, the average position (v_i) over the point p_i can be defined as equation (3). The filtering process is illustrated in figure 4.

$$v_i = \frac{\sum_{i-2}^{i+2} p_i}{5} \qquad (3)$$

ii) Locate the smoothed joints

After the smoothed skeleton is formed, the next step is to determine which points should be split. Splitting of the skeleton is accomplished by considering chord-to-point distance accumulation value. We applied chord-to-point distance accumulation proposed by Han and Poston [6] to our method. Unlike Han and Poston's method, we use the geodesic distance instead of the Euclidean distance.

The distance feature computed by Han and Poston's method is invariant with respect to rotation and translation which is a major benefit for animating a skeleton. Chord-to-point distance accumulation is the distance measurement from the line to a point in the curve segment. The interpretation of chord-to-point distance accumulation can be defined as follows.

Let L be a fixed integer value defines a line L_i from each point p_i to p_{i+L}, where $i+L$ is taken modulo N. The perpendicular distance D_{ik} is computed from L_i to the point p_k. The distance is positive if p_k is on the left-hand side of the vector (p_{i+L}- p_i), negative otherwise. Chord-to-point distance accumulation for a point p_k and a chord length L is the sum h_L of the D_{ik} as i moves from k-L to k. That is,

$$h_L(k) = \sum_{i-k-L}^{k} D_{ik} \qquad (4)$$

Since $D_{kk} = 0$, Thus,

$$h_L(k) = \sum_{i-k-L}^{k} D_{ik} = \sum_{i-k-L+1}^{k-1} D_{ik}$$

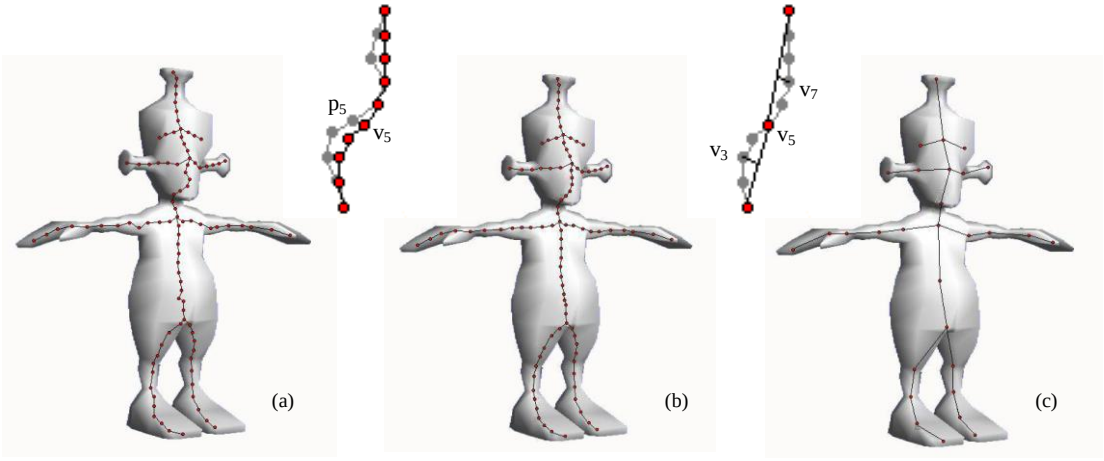

Figure 4. The method for smoothing the pruned skeleton. (a) The average position v_5 over $p_5 = (p_3+p_4+p_5+p_6+p_7)/5$, in order to obtain the smoothed skeleton, the position v_i is computed for each p_i on the pruned skeleton. (b) The result of applying the smoothing operation to the pruned skeleton from figure 4 (a), Once the smoothed skeleton is formed, chord-to-point distance accumulation is applied for splitting the smoothed skeleton into segments in order to form bones and joints. Notice that the best split is formed at v_5 since the value of chord-to-point distance accumulation at v_3 and v_7 are equal. (c) The new joints are formed as a result of the best splitting based on chord-to-point distance accumulation value.

Figure 5 illustrates chord-to-point distance accumulation, and shows an example in the case of D_{ij} is positive and D_{ik} is negative.

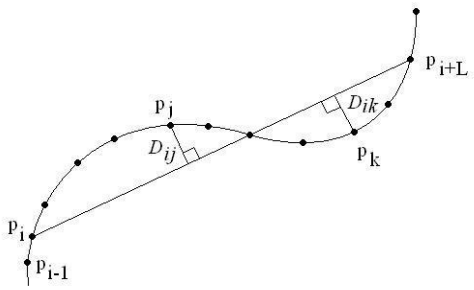

Figure 5. Chord-to-point distance accumulation.

The basic idea of locating the new joints depends on chord-to-point distance accumulation value. After the smoothed skeleton is formed, the points that align along the smoothed skeleton can be classified into three classes. The end point is a point that has only one adjacent neighbor. The junction point is a point that has three or more adjacent neighbors. The intermediate point is a point that has exactly two adjacent neighbors. The end points and the junction points split the smoothed skeleton into a set of connected segment or bones.

We use the end points and the junction points to form the initial joints of the skeleton. In each segment of the skeleton (see figure 4 (b)), we iteratively select the point from v_{i+1} to v_{N-1} and compute chord-to-point distance accumulation correspond to each selected point. The selected point becomes the splitting point if $\|D_{ij}\| = \|D_{ik}\|$ e.g. v_5 is the splitting point since chord-to-point distance accumulation value at v_3 equals to chord-to-point distance accumulation value at v_7 . Then we can form two bones from the corresponding segment, the first bone whose joints are located on v_1 and v_5, another bone whose joints are located on v_5 and v_{10} respectively. In the case of more than one splitting point is created, the sub segment will be formed (this sub segment becomes a bone later on), and more joints may be created from the corresponding sub segment. Figure 4 (c) shows the final result of the smoothed skeleton generated from our method. The algorithm for smoothing the skeleton is also shown.

```
//Given skeleton V consists of N points places in a
queue Q.
Smoothen(V, N)
  If N ≤ 0
    Return V
  Repeat
    Q ← V
    N ← 1
    Repeat while Q ≠ ϕ
      v ← POP(Q)
       Compute h_L(k)
      If ||D_ij|| = ||D_ik||
        V ← V-v
```

Algorithm 2 The iterative smoothing.

IV. Experimental Results

The pruning algorithm is tested on the Princeton Shape Benchmark database [13], and a standard handmade model (from Autodesk's Maya software). A brief comparison between the results of the proposed method and their original skeleton, extracted by using the Reeb graph is provided. The

proposed method can reduce the number of the branched connection points, which is similar to the number of critical points in the original Dijkstra's algorithm therefore the spurious skeleton joints can be removed. These joints will not influence the structure of the skeleton because they are not on any of the shortest paths computed by Dijkstra's algorithm. The pruned skeleton can then be generated in the sense that it can capture the essential shape of a 3D object while preserving a compact form of its data structure. Figure 6 shows the experimental results.

Mesh	Number of vertices	
	The Reeb graph	The pruned skeleton
Cow	2904	2655
Rabbit	1238	1026
The Dragon	1166	1011
The Alien	429	303
The Femme	635	503
Boy	17342	16017

Table 1. Comparison of the number of points between the proposed method and the original model.

Object	Number of vertexes	Number of faces	Number of control segments	
			The pruned skeleton	The smoothed skeleton
M149-boy	16017	32007	860	19
M232-alien	732	1436	40	30
Femme	503	1002	50	26
Horse	535	1058	56	27
Dinopet	2039	3999	120	31

Table 2. Comparison of the number of control segments between the smoothed skeleton and the pruned skeleton.

Table 1 compares the number of points after applying the pruning algorithm with the number of points of the original 3D objects. The proposed method can reduce the number of critical points in the graph's nodes, the branched connection points in a skeleton graph. Therefore, the numbers of the spurious skeleton joints are reduced comparable to the number of unnecessary joints generated by the Reeb graph method.

We also demonstrate our smoothing alogorithm on several pruned skeletons generated from algorithm 1, as shown in figure 7. A brief comparison between the results of the smoothing algorithm and their original pruned skeletons is provided (figure 6). The smoothing algorithm can produce a useful control skeleton, the number of control segments can be significant reduced in the sense that it is still enough for use in animation. Most of the control skeletons produced from our algorithm are centralized, and run along to the ends of the main branches of the models.

Table 2 shows the results of the number of control segments after applying the smoothing algorithm with the number of control segments of the original pruned skeletons. Figure 6 (c) shows several objects of a human-like model as smoothed by our algorithm. Segments and joints relate fairly well to surface features; however there is room for improvement. For instance for the boy-M149, there is no control points for fingers or toes are produced.

The computational complexity of our approach can be derived below.

Let n be the number of points in the meshed model. The time complexity of our algorithm can be considered as follows.

The pruning algorithm: the computation of degree of each point and the point analysis takes $O(n)$ time. The computation of testing and finding the maximal disks in the neighborhood of each point takes $O(n)$ time. The computation of the single source shortest paths (in term of the geodesic distance from that source vertex) to all points on the mesh takes $O(n \log n)$ time [5]. Thus, the overall computation takes $O(n \log n)$ time.

The smoothing algorithm: the computational complexity of averaging position of v_i over the point p_i in the filtering process takes $O(n)$ steps. The computational complexity of chord-to-point distance accumulation takes $O(n)$ steps. We implement our algorithm to place n points in a queue data structure, and most of the times are spent on the *POP* operation, thus the time complexity of this process is $O(\log n)$ steps. Thus, the overall computational complexity of the smoothing algorithm takes $O(n)$ steps (this is not include the complexity analysis of the skeleton pruning algorithm which takes $O(n \log n)$ steps).

V. Conclusion and Future Works

In this paper we have presented our approach of geodesic-based skeleton smoothing to compute the skeleton for 3D meshed model. Using Blum's Medial Axis Transform together with the geodesic distance function, and Han and Poston's Chord-to-point Distance Accumulation, we can generate one-voxel thick, graph-like skeleton which can capture the essential shape characteristics in a compact form. Our approach also maintain the robustness against any changes of a rotation and/or a translation of the 3D meshed model which are the major properties of 3D objects. In figure 6 (a) and (d), it is easy to observe that the skeleton extracted by using the Reeb graph is very sensitive to the connectivity of the boundary representation and it also has a problem in producing the unwanted skeleton joints. The pruning algorithm can overcome these drawbacks; the results are shown in figure 6 (b) and (e). In the smoothing step, the collection of points produced from our smoothing algorithm is transformed into a collection of bones which can be used for 3D animation. The main process of this method is based on chord-to-point distance accumulation introduced by Han and Poston [6]. Unlike Han and Poston's method, we use the geodesic distance instead of the Euclidean distance. We gain a benefit of the invariance

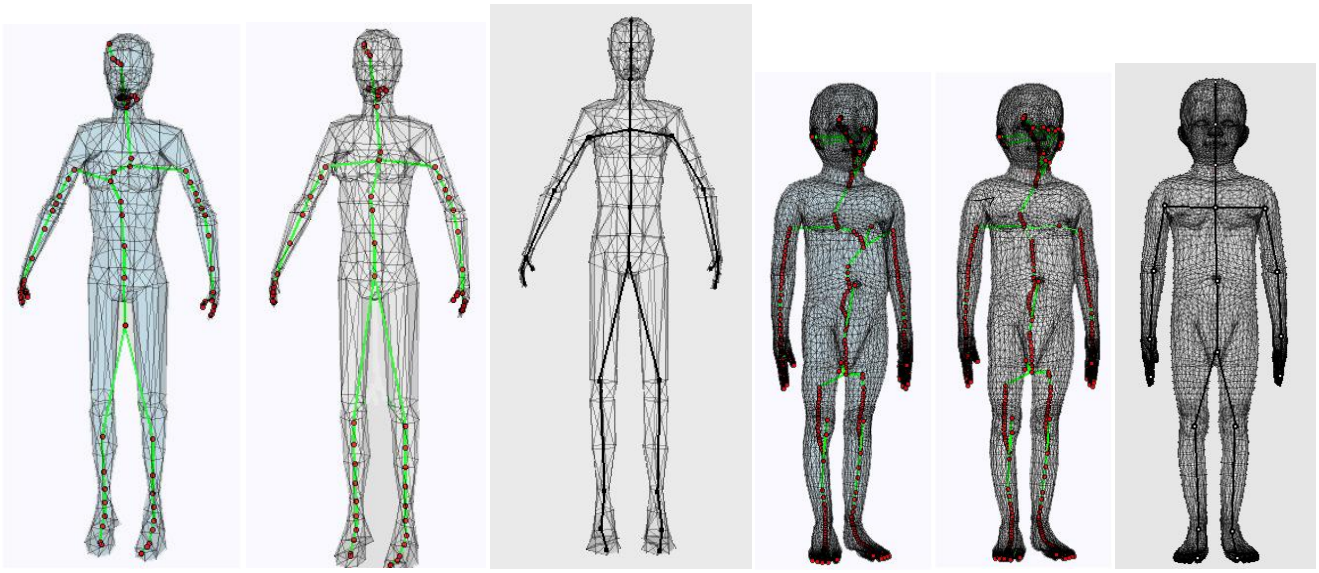

Figure 6. The pruned skeletons (b) and (e), and their skeletons extracted by using the Reeb graph (a) and (d). (c) and (f) are the results of geodesic-based skeleton smoothing.

against a rotation and a translation from the geodesic distance used in chord-to-point distance accumulation to construct the rotation and the translation invariant control skeleton. The algorithm produces the useful control skeletons with a small number of control segments. It suits for producing skeletons for more complex objects (the object which have a large number of vertex, and faces), since it can reduce the number of control segments comparable to the number of unnecessary segments generated by the skeleton pruning method. Figure 6 (c) and (f) show the meaningful characteristics of the 3D meshed model, and the generated skeletons whose joints can be associated with the 3D meshed model.

Given the skeleton construction process with the proposed approach of geodesic-based skeleton smoothing, significant improvements are desirable, either by increasing algorithm robustness or speed. Another robustness, we are taking into account is the robustness against the deformation of a model into the motion tween, which is the major technique for producing 3D character animation.

REFERENCES

[1] M. K. Agoston. Computer graphics and geometric modeling:implementation and algorithms. Springer, ISBN 1852338180, page 185, 2005.

[2] G. Aujay, F. Hetroy, F. Lazarus, and C. Depraz. Harmonic skeleton for realistic character animation. In *Symposium on Computer Animation (ACM SIGGRAPH)*, 2007.

[3] S. Baloch, H. Krim, I. Kogan, and D. Zenkov. Rotationinvariant topology coding for 2D and 3D objects usingMorse theory. In *Proceeding of IEEE International Conference on Image Processing*, ICIP 2005, pages 796-799, Sep 2005.

[4] H. Blum, R.N. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition 10*, pages 167-180, 1978.

[5] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in reeb graphs of 2-manifolds. In *Symposium on Computational Geometry*, pages 344-350, 2003.

[6] J.H. Han and T. Poston, Chord-to-point distance accumulation and planar curvature: a new approach to discrete curvature. *Pattern Recognition Letters*, vol. 22, pp. 1133-1144, 2001.

[7] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. *Computer Graphics Annual Conference Series (Proc. Of SIGGRAPH)*, pages 203– 212, 2001.

[8] F. Lazarus, and A. Verroust. Level set diagrams of polyhedral objects. In *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications*, ACM Press, pages 130–140, 1999.

[9] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM Journal of Computing 16(4)*, pages 647-668, 1987.

[10] T. Oda, Y. Itoh, W. Nakai, K. Nomura, Y. Kitamura, and F. Kishino. Interactive skeleton extraction using geodesic distance. In *Proceeding of the 16th International Conference on Artificial Reality and Telexistence*, 2006.

[11] G. Reeb. Sur les points singuliers d'une forme de pfaff completement intergrable ou d'une fonction numerique [On the singular points of a complete integral pfaff form or of a numerical function].*Comptes Rendus Acad.Science Paris 222*, pages 847–849, 1946.

[12] M. Sabry Hassouna and Aly A. Farag. On the extraction of curve skeletons using gradient vector flow. *Proc. of IEEE International Conference on Computer Vision ICCV*, Rio de Janeiro, Brazil, October 14-20, 2007.

[13] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton shape benchmark. In Shape Modeling International, pages 167-178, 2004.

[14] O. Symonova. Shape description for 3D model retrieval. Technical Report #DISI-08-010, University of Trento, Mar 2008.

[15] O. Symonova, and G. Ucelli. Using topology representation for comparison and retrieval of CAD models. *Computer Graphics topics*, pages 29-30, 5/2005, Vol. 17.

[16] P. Visutsak and K. Prachumrak. The 3D skeleton pruning for removing undesired joints. In *Proceedings of The Second IEEE International Conference on Computer, Control and Communication IEEE-IC4 2009*, Karachi, Pakistan, February 17-18, 2009.

[17] P. Visutsak and K. Prachumrak. The smoothed 3D skeleton for

animation. In *Proceedings of the 5th International Conference on INC, IDC, IMS*, Seoul, Korea, August 25-27, 2009.

[18] L. Wade and R.E. Parent. Automated generation of control skeletons for use in animation. *The Visual Computer 18*, 2002.

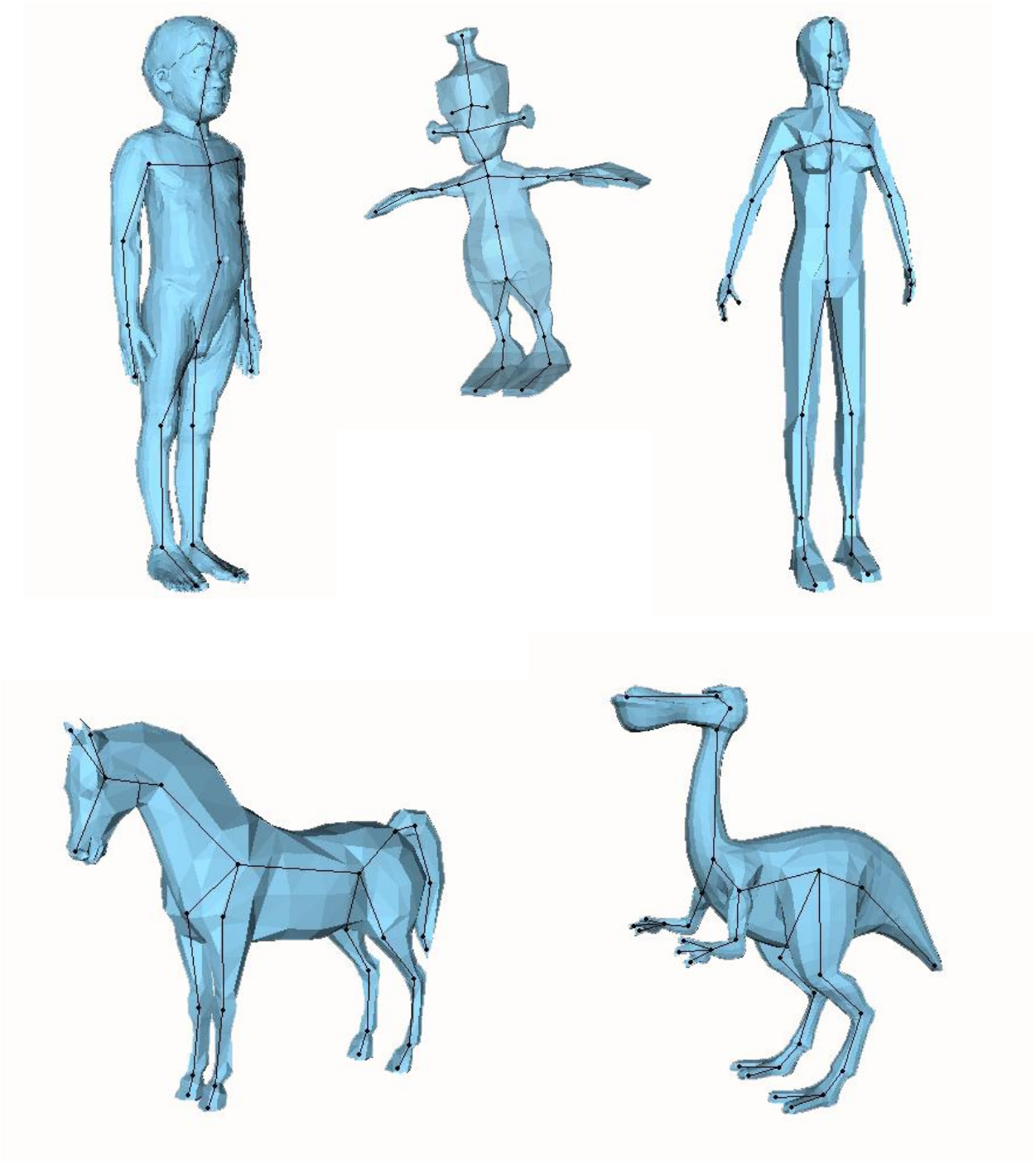

Figure 7. The smoothed skeleton of the objects: M149, M232, Femme, Horse, and Dinopet respectively.