Pacific Graphics 2003, Oct 2003, Canmore, Alberta, Canada.

Automatic Animation Skeleton Construction Using Repulsive Force Field

Pin-Chou Liu, Fu-Che Wu, Wan-Chun Ma, Rung-Huei Liang, Ming Ouhyoung
Communication and Multimedia Laboratory
Dept. of Computer Science and Information Engineering
National Taiwan University
{toby, joyce, firebird, liang}@cmlab.csie.ntu.edu.tw, ming@csie.ntu.edu.tw

Abstract

A method is proposed in this paper to automatically generate the animation skeleton of a model such that the model can be manipulated according to the skeleton. With our method, users can construct the skeleton in a short time, and bring a static model both dynamic and alive.

The primary steps of our method are finding skeleton joints, connecting the joints to form the animation skeleton, and binding skin vertices to the joints. Initially, a repulsive force field is constructed inside a given model, and a set of points with local minimal force magnitude are found based on the force field. Then, a modified thinning algorithm for volume data is applied, so that the thinning process will stop at those already found local minimum points. Referring to the topological information, we define the skeleton joints and connect them to form the skeleton. After constructing the skeleton, we anchoring skin vertices to the skeleton joints according to the distances between vertices and joints. In order to compute the repulsive force, hundreds of rays are sampled from a position inside the model, and the computation of the intersection points takes most of the time. Therefore, an octree structure is used to accelerate the joints finding process. Currently, the skeleton generated from a typical 3D model with 1000 to 10000 polygons takes less than 2 minutes on a Pentium IV 2.4 GHz PC.

1 Introduction

Because of the improvement of the computer graphics technologies, more and more artificial characters take their places in commercial industries nowadays. For example, we can see the dinosaurs chasing people in ”Jurassic Park”, and monsters scaring children in ”Monsters, Inc.”, etc. If we want to pose a 3D model, adjusting its polygon positions directly will consume a lot of time. Therefore, using the articulated model that has animation skeleton (sometimes called IK or control skeleton) will facilitate the posing process, and all the motions can be accomplished by just adjusting corresponding animation skeleton. There are many modeling and animation packages providing such functions, users can use these packages to construct an animation skeleton for a model. However, an inexperienced user usually takes hours only to construct an unacceptable result. Even an well-trained animator still requires several hours to do the job. In this paper, an automatic method is proposed to generate an animation skeleton. Initially, we build the repulsive force field inside the model, and choose the local minimum points as joint candidates. After a modified thinning algorithm is applied, we determine the final joints and generate the skeleton. Finally, we bind skin vertices to the skeleton joints.

2 Previous Work

Many previous works focused on skeleton construction, all of which produce slightly different results. Generally speaking, there are several typical approaches to extract the skeleton from an object. Intuitively, Medial Axis Transform (MAT) [5] is a typical approach to construct skeleton directly from the surface points of a 2D or 3D object [2, 11, 13]. However, the skeleton extracted from Voronoi diagram is often too noisy, and several algorithms such as the pruning or bifurcation are proposed to solve these problems [3, 6, 12, 16]. In general, the MAT skeleton is not appropriate for animation use. Many studies also try to build skeleton by constructing a discrete distance field with voxelization of the object [4, 7, 18]. After obtaining a distance field, there are many ways to obtain the skeleton from the distance field, such as thinning [10, 15] or extraction [17]. A comparatively diverse one-dimensional skeletonization method is to use the Reeb graph [8, 9]. The definition of the Reeb graph was introduced by Reeb [14]. The idea of Reeb graph skeleton is to use a continuous function, usually a geodesic distance function, to describe the topological structure and reveal the topological changes (such as merging or splitting).

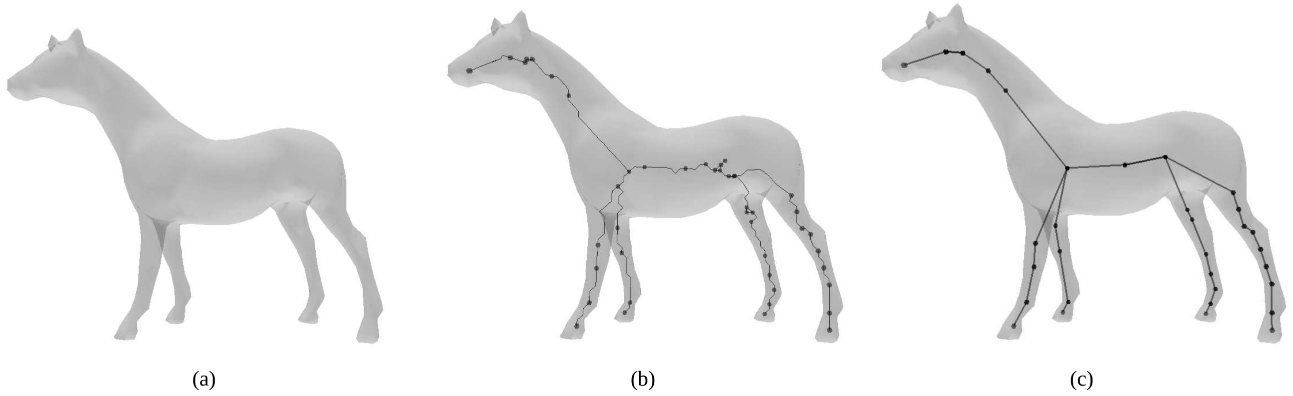

Figure 1. Illustration of the skeletonization process. (a) Original 3D model. (b) Locate local minimum points in the repulsive force field and apply the thinning method to get the initial skeleton. (c) Final result after the initial skeleton being refined.

3 Proposed Algorithm

The primary goal of our system is to automatically construct the animation skeleton for a given model. As a good automation algorithm, it only requires two primary parameters, which are *Octree-Level* and *Voxel-Size*. In general, we can generate reasonable results by default parameter settings. The input of our system is restricted to triangulated polygonal model, which may contain single or multiple individual parts. For uniformity, we scale the model initially so that its bounding box lies just inside an unit cube.

3.1 Construction of the Octree Structure

To construct the repulsive force field, we have to calculate the repulsive force for each sample location inside the model. We shoot about one hundred rays from an interior sample location and compute the intersection points of the rays with the model surface. The computation complexity for each ray is $O(n)$, where n is the polygon number of the input model, and we will reduce the complexity to the constant time. The basic idea is that we construct the octree structure for the input model, so that each leaf node owns only one or two triangles. Thus, the complexity is reduced to $O(c \times m)$, where m is the level of octree structure and c is a constant number of the triangles owned by leaf nodes. We use the projection method to assign the triangles instead of splitting them into different nodes. At first, we project each triangle to x-y, y-z, and x-z planes, and then we determine which child nodes are occupied by the projected triangle. After merging the three results, we assign the triangle to proper child nodes, and recursively repeat the process for each child node until the level of octree equals to the *Octree-Level* parameter. The construction process of a 10-level octree completes only within 2 seconds for a model which is composed of 5000 triangles. After constructing the octree structure, the performance improves about 10 times than the brute force sampling process.

3.2 Building Repulsive Force Field

A modified voxelization is used to construct the repulsive force field, with the resolution being defined by *Voxel-Size* parameter. Traditionally, we intersect the grid with the triangles, and determine the voxels as either interior or exterior voxels. We skip the grid intersecting process, and only sample the voxel center points directly. We intersect one ray from each point with the model, and determine the interior points by the intersection result. The more interior points, the better skeleton can be constructed. Experiments have shown that more than 5000 interior points can produce well results, but how to get the appropriate number of interior points still depends on the shape of the model.

Next, we shoot about one hundred rays from each interior point to compute the repulsive force, as shown in Figure 2. We use a mathematical approximation method [1] to generate the direction of these rays so that they are evenly distributed. We use s_i to represent a unit vector along the direction of ray i. Then, the repulsive force F_p at an interior point p is calculated as:

$$\overrightarrow{F_p}(x) = \sum_{i=1}^{N} f(\|r_i - p\|_2) \cdot \overrightarrow{s_i}$$

where N is the number of rays, r_i is the intersection point of a ray i on the surface. In here we take $f(x) = x^{-2}$ as the Newtonian potential function. While calculating the repulsive force, the smallest length between r_i and p is

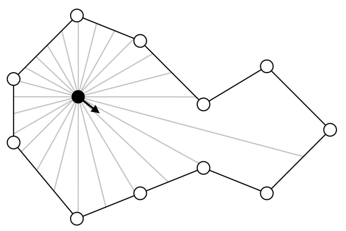

Figure 2. Rays are shot to computer the repulsive force. The gray lines represent the ray directions and the black arrow indicates the direction and magnitude of the force.

recorded for later usage. After obtaining the force field, local minimum points are located by comparing their force value with their neighbors, and we mark them as the skeleton joint candidates.

3.3 Skeletonization Process

Based on those skeleton joint candidates, we want to determine the final joints and a hierarchical relationship among the joints. First, a modified thinning algorithm for volume data is used to find the hierarchical relationship. Traditionally, the thinning algorithm removes undesired points from the surface toward inside until meets a threshold, but the results often form a surface instead of a skeleton. We modify the algorithm such that we remove points from the ones with higher force value toward the ones with lower force value until there is no more points can be removed. In addition, we define two types in which the points can't be removed:

1. The point is one of the joint candidates.
2. After the point is removed, its 26-neighbor points will be divided into two or more parts.

The first type means the thinning process will preserve the important feature parts and also prevent the result from the noise interferences. The second type means we check the local connectivity to keep the rest points being a 26-connected group during the thinning process. The reason why we don't check the global connectivity is that checking the global connectivity will create a C-shaped skeleton for a donut model, while our method will create an O-shaped skeleton. From a perceptual viewpoint, the O-shaped skeleton is more representative of the donut's topology. After the thinning process, the rest points form a rough skeleton. We add the points that have more than two neighbor points to joints candidates. Based on the connectivity among the rest points, a breadth-first search algorithm is applied to connect those candidates. However, the number of the candidates is too many to be an appropriate skeleton. We use the previous recorded shortest distance to the surface of each candidate joint to group those candidate joints, and merge their connections. Then, in each joint group, we choose the one with minimal force value to be the final joint. Figure 1 reveals the skeleton generation process.

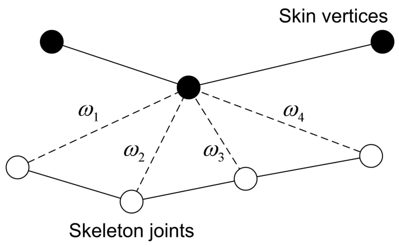

Figure 3. Binding skin vertices.

3.4 Binding Skin Vertices

Once we get a skeleton for a given model, we bind the skin vertices to the skeleton joints in order to deform the model. The corresponding relationship between a skin vertex and all joints is calculated as:

$$T_i = \sum_{j=1}^{N} \omega_j \times t_j$$

where, T_i is the transformation matrix of a skin vertex, N is the number of the joints, and t_j is the transformation matrix of a joint with a weight w_j. When the skeleton is moved, the transformation matrices of the joints will be applied to the skin vertices with this weighting function. There are two types of skin vertices binding. If we attach a rigid model, such as a table or a robot, to its skeleton, the movement of its skin is expected to be inflexible. Therefore, we anchor a skin vertex to one nearest joint, and the weight of the nearest joint is set to one. On the contrary, if we attach an animal or a human to his skeleton, we expect the vertices to transform smoothly and vitally. Conventionally, as Figure 3 shows, a skin vertex is anchored to the nearby joints and we compute the weights for these joints as:

$$\omega_j = \frac{D - d_j}{(S-1) \times D}$$

where S is the number of nearest joints (equals to four in this case), d_j is the distance between a nearest joint and a skin vertex, and D is the sum of all d_j. Our method doesn't transform the coordinates of skin vertices into the local coordinates of skeleton joints, and it is compatible with many 3D software or libraries, such as Maya, 3D Studio Max, Direct3D, and so on.

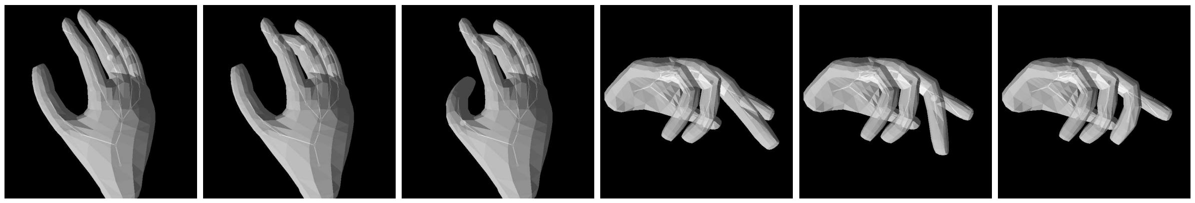

Figure 4. Deformation sequence of a hand model.

4 Results

Several generated skeletons are shown in Figure 6. Generally speaking, our method can generate the skeleton of a given model in few minutes. In most cases, we can produce reasonably good results within 2 minutes. The generated skeleton conforms to the topology of the model, as shown in Figure 5. It also represents primary features of the model so that they are suitable for animation. Figure 4 shows the animation sequence of a hand model which is driven by the generated skeleton.

5 Conclusions and Future Work

An effective algorithm is proposed to facilitate the skeleton-constructing process which helps animators to speed up their jobs. Furthermore, this method achieves a higher degree of automation than previous methods, and it produce the animation skeleton of relatively good quality with little user inputs. We use the repulsive force instead of the distance value, so that we can prevent our algorithm from ambiguous joints. Therefore, we can produce more consistent results in various models. For example, we produce only one joint in disk-like model, while using distance value will produce many candidates with same values.

At present, the resolution of the voxel representation is fixed now and some tiny details of the model may be ignored. For that reason, it is important to involve adaptive voxelization. Then, using different resolutions for different parts will provide enough information to determine more precise joints.

References

[1] http://www.math.niu.edu/ rusin/known-math/95/sphere.faq.

[2] N. Amenta, S. Choi, and R. Kolluri. The power crust. *ACM Symposium on Solid Modeling and Applications*, pages 249–260, 2001.

[3] D. Attali and A. Montanvert. Modeling noise for a better simplification of skeletons. *IEEE International Conference on Image Processing*, pages 13–16, 1996.

[4] I. Bitter, A. E. Kaufman, and M. Sato. Penalized-distance volumetric skeleton algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):195–206, 2001.

[5] H. Blum. *A Transformation for Extracting New Descriptors of Shape*, pages 362–380. MIT Press, 1967.

[6] S. W. Choi and H. P. Seidel. One-sided stability of medial axis transform. *Lecture Notes in Computer Science 2191*, pages 132–139, 2001.

[7] N. Gagvani, D. Kenchammana-Hosekote, and D. Silver. Volume animation using the skeleton tree. *IEEE Symposium on Volume Visualization*, pages 47–54, 1998.

[8] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. *SIGGRAPH 2001 Conference Proceedings*, pages 203–212.

[9] F. Lazarus and A. Verroust. Level set diagrams of polyhedral objects. *Proceedings of the fifth ACM symposium on Solid modeling and applications*, pages 130–140, 1999.

[10] T.-C. Lee, R. L. Kashyap, and C.-N. Chu. Building skeleton models via 3-d medial surface/axis thinning algorithms. *Computer Vision, Graphics, and Image Processing*, 56(6):462–478, 1994.

[11] N. Mayya and V. T. Rajan. Voronoi diagrams of polygons: A framework for shape representation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 638–643, 1994.

[12] R. Ogniewicz. Automatic medial axis pruning by mapping characteristics of boundaries evolving under the euclidean geometric heat flow onto voronoi skeletons. *Technical Report 95-4, Harvard Robotics Laboratory*, 1995.

[13] R. Ogniewicz and M. Ilg. Voronoi skeletons: Theory and applications. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 63–69, 1992.

[14] G. Reeb. Sur les points singuliers d’une forme de pfaff completement integrable ou d’une fonction numerique. *Comptes Rendus Acad. Science Paris*, 222:847–849, 1946.

[15] R. C. Staunton. An analysis of hexagonal thinning algorithms and skeletal shape representation. *Pattern Recognition*, 29(7):1131–1146, 1996.

[16] R. Tam and W. Heidrich. Feature-preserving medial axis noise removal. *European Conference on Computer Vision*, pages 672–686, 2002.

[17] L. Wade and R. E. Parent. Automated generation of control skeletons for use in animation. *The Visual Computer*, 18(2):97–110, 2002.

[18] Y. Zhou and A. Toga. Efficient skeletonization of volumetric objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):195–206, 1999.

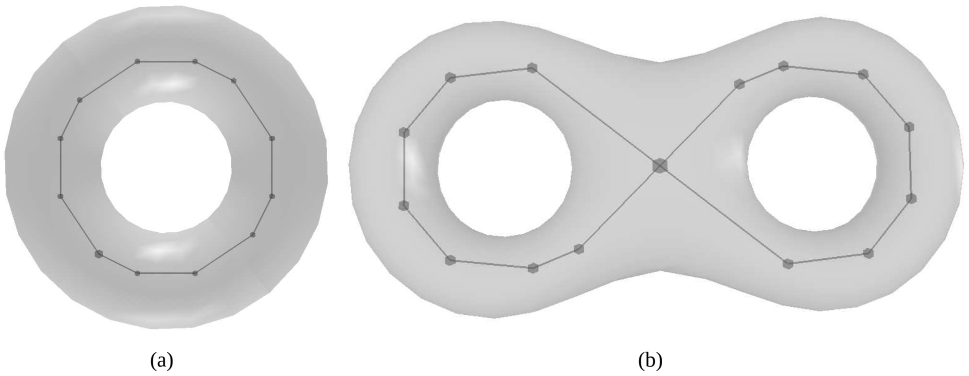

Figure 5. Genus. The generated skeleton conforms to the topology of the model.

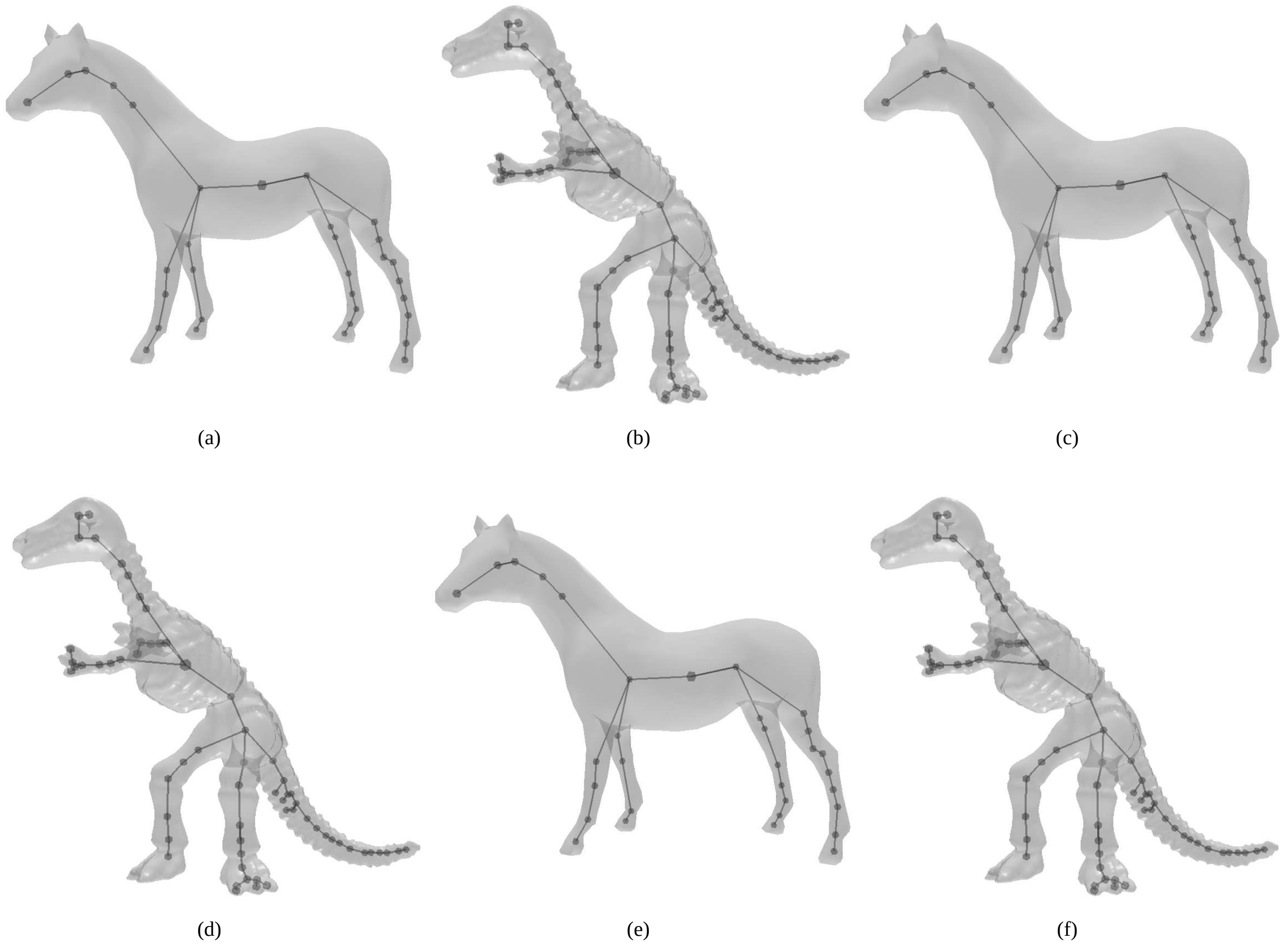

Figure 6. Result. Generally speaking, our method can generate the skeleton of a given model in few minutes. In most cases, we can produce reasonably good results within 2 minutes.