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 an animation skeleton, and binding skin vertices to the skeleton. 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 is applied to generate an initial skeleton, which is further refined to become the final result. When the skeleton construction completes, skin vertices are anchored to the skeleton joints according to the distances between the vertices and joints. In order to build the repulsive force field, hundreds of rays are shot radially from positions inside the model, and it leads to that the force field computation takes most of the execution time. Therefore, an octree structure is used to accelerate this process. Currently, the skeleton generated from a typical 3D model with 1000 to 10000 polygons takes less than 2 minutes on a Intel Pentium 4 2.4 GHz PC.

1 Introduction

Because of the improvement of the computer graphics technologies, more and more virtual characters take their places in commercial industries. For example, we can see dinosaurs chasing people in "Jurassic Park", and monsters scaring children in "Monsters, Inc.", etc. The need of computer animation becomes an essential issue nowadays. Posing is always an important topic in computer animation. It is impossible to pose a 3D model by changing every polygon positions directly. Therefore, using the articulated model that has an animation skeleton (sometimes called Inverse-Kinematics or control skeleton) will facilitate the posing process, and all the movements can be accomplished by just adjusting corresponding parts of the animation skeleton. There are many animation packages which provide such functionalities for users to construct animation skeletons for 3D models. However, these packages are complex and difficult for an inexperienced user, even a well-trained animator still requires several hours to finish his job.

In this paper, an automation algorithm is proposed to generate animation skeletons from models. Initially, we build the repulsive force field by shooting rays from sample points inside the model to compute the force magnitude. The repulsive force we used here is similar to electro-static force, which leads to an interesting phenomenon: electromagnetic shielding effectiveness. It stands for that the model surfaces which are invisible from a sample point do not contribute to the repulsive force. We choose the points with local minimal force magnitude as joint candidates. After a modified thinning algorithm being applied, we determine the final joints and generate the skeleton. Finally, skin vertices are bound to the skeleton joints such that we can use the skeleton to pose the input model.

2 Previous Work

Intuitively, Medial Axis Transform (MAT) [5] is a typical approach to construct skeleton directly from the Voronoi diagram of the object surface points [2, 11, 13]. However, the MAT skeleton is often too noisy, and several algorithms such as pruning or bifurcation are proposed to solve this problem [3, 6, 12, 16]. In general, the MAT skeleton is not appropriate for animation use. People also try to build skeletons by constructing a discrete distance field by means of the voxelization of the object [4, 7, 18]. After obtaining a distance field, one may use methods such as thinning [10, 15] or extraction [17] to get the skeleton. A comparatively diverse method is to use the Reeb graph [8, 9, 14]. This method produces good one-dimensional skeletons which fit the shapes of the 3D models.

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 parameters, which are *Octree-Level* and *Voxel-Size*. Mostly, we can generate reasonable results by default settings. The input of our system is restricted to triangulated polygonal models, 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 Building the Repulsive Force Field

A modified voxelization is used to construct the repulsive force field, with the resolution being defined by *Voxel-Size* parameter. We shoot one ray from the center point of each voxel to check if the point is interior or exterior of the model. The more interior points, the better skeleton can be constructed. However, how to get the appropriate number of interior points still depends on the shape of the model. Next, about 100 rays are shot radially from each interior point to compute the repulsive force, as shown in Figure 1. We use a mathematical 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_r at an interior point p is calculated as:

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

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 recorded for later use.

3.2 Constructing the Octree Structure

To speed up the repulsive force field construction process, an octree structure is introduced to our system. The original time complexity for each ray intersection is $O(n)$, where n is the polygon number of the input model. We now use the octree structure to reduce the complexity to the constant time. The basic idea is that we construct an 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 a 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 the octree structure being built, the performance improves about 10 times than the brute force sampling process.

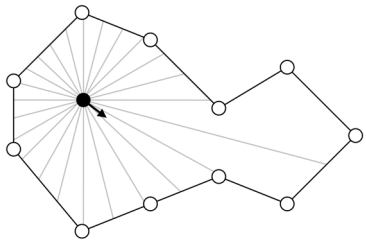


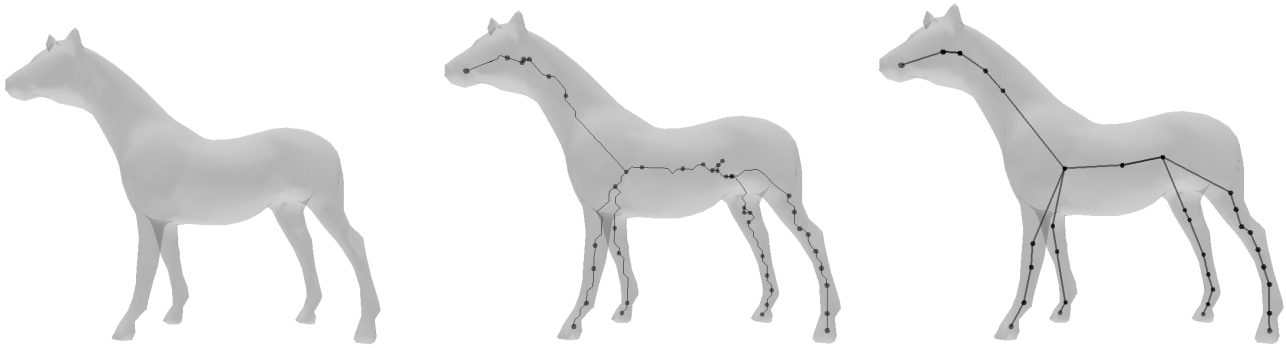
Figure 1. Rays are shot radially to compute the repulsive force. The gray lines represent the ray directions and the black arrow indicates the direction and magnitude of the force.

3.3 Skeleton Construction Process

After obtaining the force field, for each point we compare its force magnitude with the 26 neighbors. We mark the local minimum points as the skeleton joint candidates. Based on these 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. A traditional thinning algorithm removes undesired points from the surface toward inside until a threshold is reached, but the result often forms a medial 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 rules by which the points can't be removed:

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

The first rule ensures that the thinning process not only preserve the important feature parts but prevent the result from the noise interference. The second rule checks the local connectivity to keep the rest points being 26-connected 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,



(a) (b) (c)

Figure 2. Illustration of the skeleton construction 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.

while our method will create an O-shaped skeleton. From a mathematical viewpoint, the O-shaped skeleton conforms more to the topology of the donut.

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, sometimes the number of the candidates is too large such that it is hard to form 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 group, we choose the one with minimal force value to be the final joint. Figure 2 reveals the skeleton construction process.

3.4 Binding Skin Vertices

Once we get the skeleton, we bind the skin vertices to its joints. The 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 i, N is the number of the nearby joints, and t_j is the transformation matrix of a joint with a weight w_j. When the skeleton is adjusted, 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 supposeed to be inflexible. Therefore, it is easy that we can only anchor a skin vertex to one nearest joint, and the weight of this nearest joint is set to 1. On the contrary, if we attach an animal or a human to his skeleton, we expect the vertices to transform smoothly and vividly. Conventionally, as Figure 3 shows, a skin vertex needs to be anchored to several nearby joints and we compute the weights for these joints as:

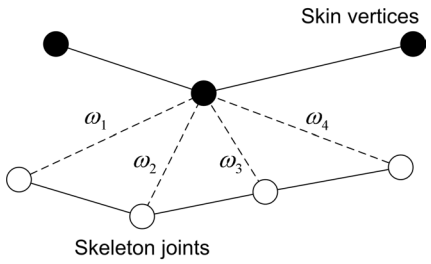


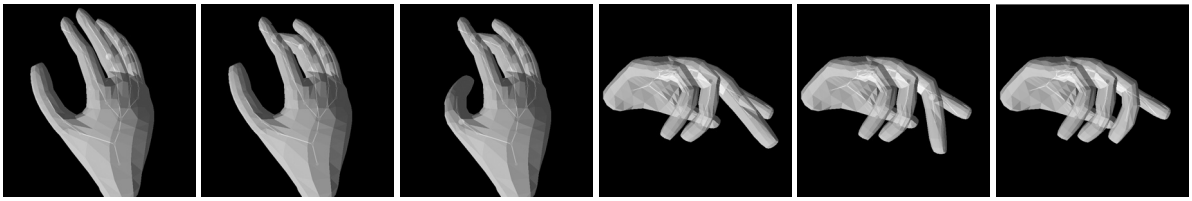
Figure 3. Binding skin vertices.

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

where S is the number of nearest joints (equals to 4 in this case), d_j is the distance between the skin vertex and a nearest joint, 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.

4 Results

Averagely, we can produce reasonably good results from 3D models which contain around 10000 triangles within 2 minutes. In most cases, the method generates the skeleton in few minutes. Figure 4 shows the animation sequence of a



(a) (b) (c) (d) (e) (f)

Figure 4. Animation sequence of a hand model.

hand model which is driven by the generated skeleton. The skeleton preserves primary features of the model which are very important for modelling and rigging. It also conforms to the topology of the model, as shown in Figure 5. More skeleton results are shown in Figure 6 in color.

5 Conclusions and Future Work

An robust algorithm is proposed to facilitate the skeleton construction process and help animators to speed up their jobs. Furthermore, this algorithm 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 Euclidean distance value, so that we can prevent our algorithm from generating ambiguous joints. Thus, the skeleton results are more consistent. For example, the algorithm produces only one joint from a disk-like model, while the other methods which use distance value may produce many candidates with same values.

At present, the resolution of the voxelization is fixed and tiny details of the model may be disregarded. For that reason, it is important to use a multi-resolution voxelization to determine more precise joint positions.

6 Acknowledgements

This work was partially supported by grants from National Science Council, Taiwan, R.O.C. under NSC91-2213-E-002-066 and Silicon Integrated Systems (SiS) Education Foundation.

References

[1] Topics on Sphere Distributions. *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. *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.

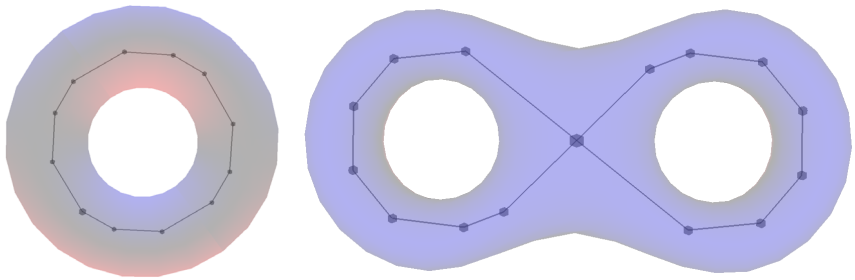
[14] G. Reeb. Sur les points singuliers d'une forme de Pfaff completement integrable ou d'une fonction numerique. *Comptes Rendus Acad. Science Paris*, 222:847–849, 1946.

[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.

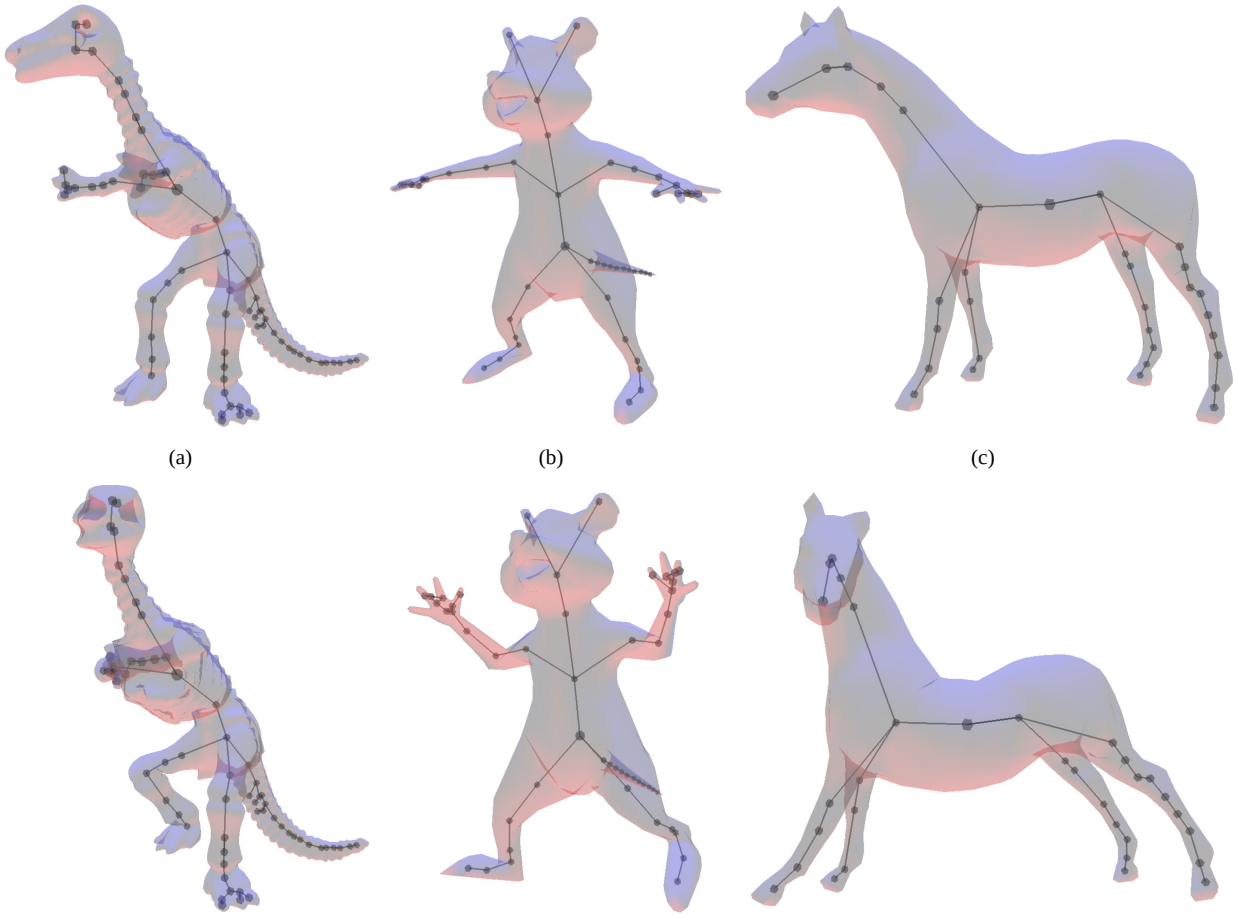
[17] L. Wade and R. E. Parent. Automated Generation of Control Skeletons for Use in Animation. *The Visual Computer*, 18(2):97–110, 2002.

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



(a) (b)

Figure 5. The generated skeleton conforms to the topology of the model. (a) A gunes-1 model and its skeleton. (b) A gunes-2 model and its skeleton.



(a) (b) (c)

(d) (e) (f)

Figure 6. Three skeletons and their different posing results. Our algorithm only requires two parameters, which are *Octree-Level* and *Voxel-Size*. Averagely, we can produce reasonably good results of a model with 10000 polygons within 2 minutes.