A Method of Automatic Cage Generation for Shape Deformation by Using Elastic Models

Takayuki Kanaya[1(✉)], Yuta Muraki[2], Koji Nishio[2], and Kenichi Kobori[2]

[1] Hiroshima International University, Hiroshima, Japan
t-kanaya@hw.hirokoku-u.ac.jp
[2] Osaka Institute of Technology, Osaka, Japan
muraki@is.oit.ac.jp,
{koji.a.nishio,kenichi.kobori}@oit.ac.jp

Abstract. Laplacian-based mesh processing technique is a kind of shape deformation in Computer Graphics modeling. It is hard to deform shapes which constructed by a lot of vertices in real time, because computational cost is high. A cages-based mesh deformation method is used in order to control the computational cost. A cage is a polyhedron which envelops an original dense model and is constructed by few meshes. The main advantages of using cages in shape deformations are controlling high speed computation. Currently, the coarse cage is constructed mainly by hand, and the construction usually takes several hours, even longer. Furthermore, when the shape of the model to be deformed is complex, it is very hard to construct its coarse cage by hand. Therefore it is important to develop a convenient method to generate the coarse cage enveloping a model. In this paper, we propose a method of automatic cage generation.

Keywords: Cage generation · Shape deformation · Computer graphics

1 Introduction

Mesh deformation is a common process in geometry modeling and computer animation. Geometry modeling techniques have become an important key technology in the industrial design and development process. Similarly, in computer animation we may want a realistic behavior, simulating physics. Therefore, we need flexible tools for mesh deformation to achieve the desired results easily. In the past years there have been many methods, such as Free-form deformation (FFD) [1], Radial Basis Functions (RBF) [2], curve based deformation [3], skeletons [4, 5] and physics simulation [6].

Also, in recent years many discrete deformation techniques based on Laplacian mesh representation have been published [7–9]. They can support interactive work by encoding differential properties and positional constraints in a linear system. Laplacian operators are described by the uniform weighting method. This approach is numerically stable, but is not able to compute in real-time, depending on the number of constraints. Especially, modern computer graphics models are often created or acquired at a very

C. Stephanidis (Ed.): HCII 2015 Posters, Part I, CCIS 528, pp. 690–695, 2015.
DOI: 10.1007/978-3-319-21380-4_116

high resolution in order to maintain a convincing level of realism. The huge number of vertices makes direct manipulation of the models tedious and computationally expensive. We would like a deformation method able to work in real-time, for interactive applications, which limits the computation time. Also, it is desirable to have a convenient and easy to use system which makes it simple for users to get quickly familiar with it. A way to reduce computational cost is to build a similar but coarse structure with fewer vertices, and then deform the dense model through the coarser structure. This coarser structure, which envelops the original dense model, is called cage.

The first approach cage based method of Laplacian mesh deformation comes from Floater et al. [10] and Ju et al. [11]. Both studies aimed at looking for an interpolation method for surfaces. Initially, Floater presented a 2D mean value coordinates (MVC) over quasi-convex polygons, then developed a 3D version of the algorithm. Then, Ju et al. developed another solution by using the Floater's 2D approach. Next, Joshi et al. [12] developed another method to set harmonic coordinates (HC) successfully avoiding some of the MVC drawbacks. Also, Lipman et al. [13] developed a method with cage faces data in deformation operators to achieve a natural deformation with shape preserving.

In cage based methods, a coarse cage enveloping a model is required to be constructed in advance for deforming the model. Currently, the coarse cage is constructed mainly by hand [10–13], and the construction usually takes several hours, even longer. Furthermore, when the shape of the model to be deformed is complex, it is very hard to construct its coarse cage by hand. Therefore it is important to develop a convenient method to generate the coarse cage enveloping a model.

2 Cage Generation Algorithm

First, we outline our method. The main steps of our method are as follows:

1. Compute the Bounding Volume (BV) of the dense mesh, and then Construct BV Tree by partitioning the BV recursively.
2. Generate the Convex Hull of a set of vertices constructing all BVs.
3. Subdivide the triangles of Convex Hull
4. Elasticize each edge of triangular net

2.1 Bounding Volumes Tree Generation

The first step of our algorithm is to compute the bounding volume (BV) of the original dense mesh model. In general, the idea is for the BV to have cheaper overlap tests than the complex objects they bound [14]. The properties for the BV include tight fitting, using little memory, encapsulating objects, and so on.

There are many kinds of the BV, such as Sphere, AABB (Axis-Aligned Bounding Box), OBB (Oriented Bounding Box), and Convex Hull [14]. We use the AABB and the OBB, as shown in Fig. 1(b). The AABB is a rectangular six-sided box categorized by having its faces oriented in such a way that its normal are at all times parallel with

the axes of the given coordinate system. The OBB is a rectangular block, like an AABB but with an arbitrary orientation. The arbitrary orientation is computed by principal component analysis, and so on.

After computing BV, we generate Bounding Volumes Tree (BVT). The BVT is a tree hierarchy which is generated by partitioning the BV recursively [14]. The original set of BV forms the leaf nodes of the tree that is this BVT. We describe the way of the BVT. (1) We compute the centroid of the original dense model. (2) We select the normal of the plane partitioning the space it is in into infinite sets of points on either side of the plane. (3) We partition the space by the plane passing through the centroid with the normal. (4) We construct 2 BVs by using the vertices in each halfspace. For example of AABB, it is possible to partition it by X-Y, Y-Z, and Z-X plane, respectively. Also, it is possible to partition it into 8 parts (Octree). A partitioned example is illustrated in Fig. 1(c).

2.2 Convex Hull Generation

The next step of our algorithm is to generate convex hull by using the vertices constructing BVT. It is so quickly and easy to generate the convex hull that the number of vertices constructing each BV is 8. We use Barber method [15] to generate the convex hull. We show an example in Fig. 1(d).

2.3 Triangles Subdivision

The convex hull, which is generated in Sect. 2.2, is consisted of triangles. We subdivide these triangles, if necessary. The midpoint is inserted into each edge constructing triangles, and then we generate triangular net by subdividing triangles in quarter. A subdivided example is illustrated in Fig. 1(e).

However, several warping triangles occur, when they are subdivided. Then we remesh the triangles, if necessary.

2.4 Elasticizing Each Edge of the Triangular Net

We regard the triangular net as the net with elastic force, contract the triangular net. A vertex of triangular net is moved to the position which the energy of elastic force minimizes. Similarly, all vertices are translated iteratively until the total energy minimizes. As a result, we generate a Cage.

2.4.1 Definition of Energy

We describe the triangular net with elastic force. All vertices are connected to each other with spring. We define the energy of each vertex as follows:

$$E(x) = \alpha \sum_{i=0}^{5} \left[\max_{v_j} \left\{ \left(p(v_j) - p(x) \right) \bullet e_i \right\} \right]^2 \quad (1)$$

where, $p(x)$ is a candidate vertex, $p(v_j)$ is each vertex connecting $p(x)$, α is spring constant, e_i is unit vector along each positive or negative axis.

We describe the contracting procedure as follows:

1. Select a candidate vertex x among vertices constructing triangular net.
2. Determine a new position which the energy minimizes when the vertex x moves very short distance from the current position to 26-adjacency, respectively.
3. Go to the step 1, until all vertices determine their new positions.
4. Move all vertices to their new positions, respectively.
5. Repeat step 1 to 4, until the all energy drops below a threshold value.

2.4.2 Collision Detection

A vertex x happens to intersect with the original model, when the x moves to the new position decided in Sect. 2.4.1. This case violates a fundamental rule that the cage includes the original model. Therefore we have to avoid intersecting with it.

In the case of intersecting of edges constructing the triangular net with faces consisting of the original model according to moving the x to the new position, we do not allow the x to move.

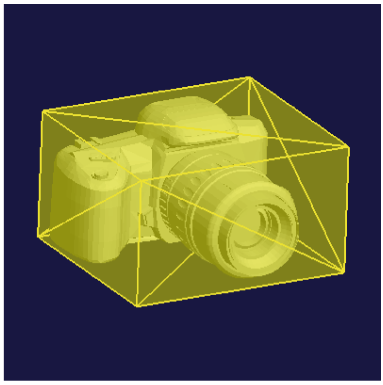
3 Results

Our algorithm developed in this paper is implemented with VC ++ 2010 and OpenGL, and runs on the PC with 3.4 GHz Core i7, 4.0 GB memory, and Windows 7 Professional (32bit). A result is illustrated in Fig. 1(a)−(f). The number of faces of the original dense model, is shown in Fig. 1(a), is 11,794. In this example which is shown in Fig. 1(b), we selected the AABB as the BV. Figure 1(c) is the BVT which is partitioned the BV into 2 parts along the Y axis twice, and then along the X axis twice, recursively. A convex hull, is shown in Fig. 1(d), was generated, and was subdivided. Next, the triangular net was transformed from the convex hull. We got the Cage according to contracting the triangular net as shown in Fig. 1(f). The number of the faces of the Cage is 344.

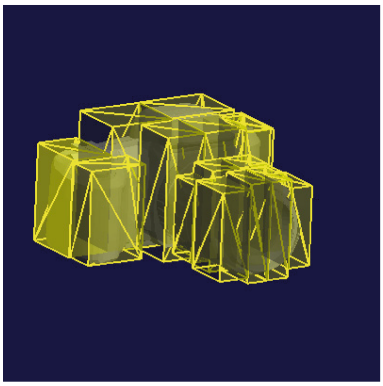
4 Conclusions

In this paper, we propose an automatic method to generate the coarse cage for a given original dense mesh. The original mesh is first enclosed the bounding volume, and the bounding volume is partitioned into 2 parts or 8 parts, recursively. Next, convex hull is generated by using the vertices constructing the bounding volumes. The convex hull is consisted of triangles. These triangles are subdivided and remeshed, if necessary. The triangular net with elastic force is contracted by moving the vertices to their new positions until the total energy minimizes. Finally, we generate a cage.

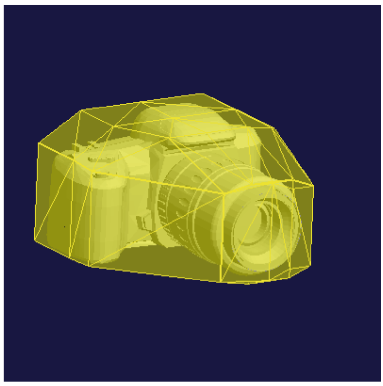
(a) An original dense model.



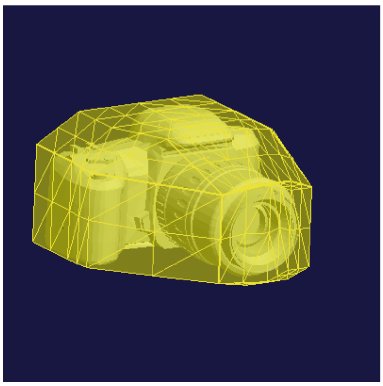
(b) A bounding volume.



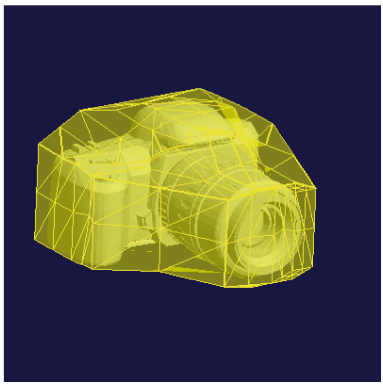
(c) A bounding volume tree.



(d) A convex hull.



(e) A triangular net with elastic force.



(f) A Cage.

Fig. 1. An example of our method

References

1. Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. SIGGRAPH Comput. Graph. **20**(4), 151–160 (1986)
2. Botsch, M., Kobbelt, L.: Real-time shape editing using radial basis functions. Comput. Graph. Forum **24**(3), 611–621 (2005)
3. Peng, Q., Jin, X., Feng, J.: Arc-length-based axial deformation and length preserved animation. In: Computer Animation 1997, pp. 86–92 (1997)
4. Yoshizawa, S., Belyaev, A.G., Seidel, H.-P.: Skeleton-based variational mesh deformations. Comput. Graph. Forum **26**(3), 255–264 (2007)
5. Yan, H.-B., Shi-Min, H., Martin, R.R., Yong-Liang, Y.: Shape Deformation Using a Skeleton to Drive Simplex Transformation. IEEE Trans. Visual. Comput. Graph. **14**(3), 693–706 (2008)
6. Terzopoulos, D., Platt, J., Barr, A., Fleischer, K.: Elastically deformable models. In: Proceedings of the 14th Annual Conference On Computer Graphics and Interactive Techniques, SIGGRAPH 1987, pp. 205–214 (1987)
7. Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H.-P.: Laplacian surface editing. In: 2004 Eurographics Symposium on Geometry Processing 2004, pp.179 −188 (2004)
8. Sorkine, O.: Laplacian mesh processing. In: STAR Proceedings of Eurographics 2005, pp. 53–70 (2005)
9. Nealen, A., Sorkine, O., Alexa, M., Cohen-Or, D.: A sketch-based interface for detail-preserving mesh editing. ACM Trans. Graph. **24**(3), 1142–1147 (2005)
10. Floater, M.S., K´os, G., Reimers, M.: 3D Mean value coordinates. Comput. Aided Geom. Des. **22**(7), 623–631 (2005)
11. Ju, T., Schaefer, S.,Warren, J.: Mean value coordinates for closed triangular meshes. In: ACM SIGGRAPH 2005, pp. 561–566 (2005)
12. Joshi, P., Meyer, M., DeRose, T., Green, B., Sanocki, T.: Harmonic coordinates for character articulation. ACM Trans. Graph. **26**(3), 1–9 (2007)
13. Lipman, Y., Levin, D., Cohen-Or, D.: Green coordinates. ACM Trans. Graph **27**, 78: 1–78:10 (2008)
14. Ericson, C.: Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology). CRC Press, Boca Raton (2004)
15. Bradford, C.B., Dobkin, D.P., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Trans. Graph. on Math. Softw. **22**(4), 469–483 (1996)