Interactive Physically-based Animation System for Dense Meshes

Ryo Kondo and Takashi Kanai

Keio University SFC, Fujisawa, Kanagawa, Japan

Abstract

In this paper we describe an interactive physically-based animation system for dense meshes. Our method extracts a coarse mesh from an original mesh to make a tetrahedral mesh for the reduction of computational costs. For computing reaction forces we precompute penetration depth values and gradients at mesh vertices by creating a distance field. They are interpolated when collisions are detected and are used for the calculation of forces with a penalty method. Our method can handle dense meshes with physically-based animation and collision response at interactive frame rates.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

Physically-based animation is essential to produce realistic scenes. An interactive animation system including collision response is, however, still problematic for deformable objects because of the complexity of collision detection and the computation of reaction forces. In this paper we provide a unified framework and a simple solution to physically-based animation for dense meshes.

Our basic idea is to use a coarse tetrahedral mesh to animate a given dense mesh interactively. The computational cost to solve linear dynamic equations is dramatically decreased and depends only on the number of elements of a tetrahedral mesh, not the complexity of a given dense mesh.

Our approach also keeps high-resolution quality of rendering. Before animation we create a relationship between a tetrahedral mesh and a dense mesh. In each frame positions of a dense mesh are updated by barycentric interpolation. As a result a natural-looking animation can be established with high quality rendering interactively.

Collision response is indispensable for realistic animation. After the collision detection we calculate reaction forces directly from a tetrahedral mesh and reflect them on the animation. We can integrate these methods into our framework to animate scenes with several dense meshes interactively.

The rest of the paper is organized as follows. Section 2 covers related work for physically-based animation and collision detection. In Section 3 we describe our framework and a post-process technique to recover the relationship between a tetrahedral mesh and a given mesh. Section 4 details the sequences for handling collisions. Results and discussions are shown in Section 5. In Section 6 we conclude and discuss about future work.

2. Related work

Physically based modeling is one of the most challenging research areas in computer graphics. In this section we focus on deformable object modeling and describe related work.

Various methods have been developed to represent a deformable object. One of the most common techniques is a finite element method [Bat82], which consider an object as finite tetrahedral (or hexahedral) elements and discretize. For vertices of these elements, we solve dynamic equation and compute their displacements. In other way a boundary element method [JP99, JP03], which requires discretization only on boundary points, and a particle method [YSY01], which represent a solid object as a cluster of particles, are well-known approaches. In most of them an object is subdivided into smaller pieces. On the other hand, some approaches such as free-form deformation (FFD) [TS86] deform an object by its surrounded coarser object with parametric representation. Faloutsos et al. [FvdPT97] proposed a combined method with FFD and physics.

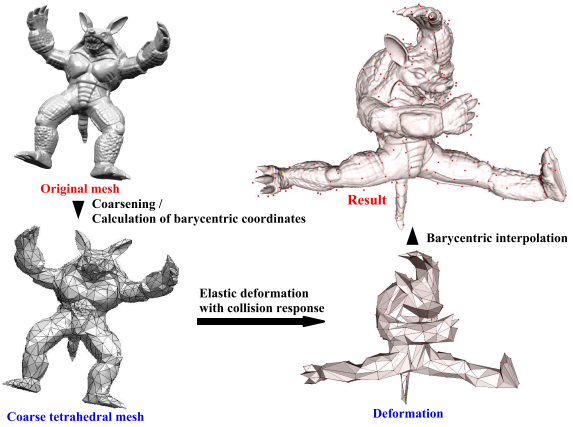

Figure 1: *Overview of our animation framework*

Stable real-time deformations [MDM*02] is an interactive approach based on linear finite element method. By rotating stiffness matrices this approach provides both fast and stable simulation. Capell et al. [CGC*02] proposed a framework for skeleton-driven deformations with adaptive FEM. Hauser et al. [HSO03] extended a modal analysis approach [PW89] for interactive physically-based animation to improve the computational cost.

This paper describes a simple unified framework for physically-based animation with dense meshes. We especially show details of using a coarse tetrahedral mesh for deformations. In addition, we integrate collision detection and reaction force computations for deformable objects.

3. Framework and mesh mapping

Figure 1 shows an overview of our framework. Our approach consists of three phases: tetrahedral mesh generation, animation with collisions and update of a dense mesh. In this section we explain the first and the third phase. The second phase, methods for animation and collision detection, are covered in the next section.

3.1. Tetrahedral mesh generation

We use [MDM*02] for physically-based elastic deformation. Note that our framework can also be used for other types of FEM algorithms. In [MDM*02], a tetrahedral mesh is needed for the deformation. To keep the implementation simple we generate an intermediate mesh, a coarse surface mesh, and create a tetrahedral mesh from this surface mesh. Since these processes are independent from our framework, external programs can be helpful. We use a Garland's mesh simplification algorithm [GH97] to generate a coarse surface mesh. This allows users to control the numbers of polygons of a coarse mesh and to apply feature-preserving mesh simplification. Usually, an initial dense surface mesh is coarsened to less than a thousand polygon.

We next create a tetrahedral mesh from a coarse surface mesh. A public tetrahedral mesh generation software, *Netgen* [Sch97], is used for this process. The number of tetrahedral elements is adjustable and can be decreased by several hundreds.

3.2. A relationship between two meshes

As explained in the previous subsection we can obtain a coarse tetrahedral mesh from a given dense mesh. However, the mesh details are removed. A relationship between these two meshes is used to restore the quality of a given mesh.

We make correspondences between vertices of a given dense mesh and elements of a tetrahedral mesh. Each vertex of an original mesh should belong to an element of the tetrahedral mesh. Once correspondences are determined new positions of vertices of a dense mesh can be updated using barycentric interpolation.

A concrete approach is as follows. Firstly, we perform an inside-outside test for each vertex of a given mesh against all the tetrahedral elements. If a vertex lies in a tetrahedral element, it belongs to this element.

For the vertices that are outside of all the tetrahedral elements, we next determine their corresponding tetrahedron. For each of such vertices we choose a nearest face which constitutes the surface of a tetrahedral mesh. Then a tetrahedral element in which a face is included is determined as a corresponding tetrahedron. After these processes each vertex of a given mesh has it's corresponding tetrahedron and its barycentric coordinate respect to the element (see also Section 4.2).

4. Collision handling

In this section we describe the process for handling collisions. By combining the following two methods we can provide a simple solution for interactive computation of the collision response. One is deformed distance fields method [FL01], which precomputes penetration depth values and gradients inside the mesh. Another one is spatial hashing method [THM*03], which can detect collisions between tetrahedral elements and vertices of other tetrahedral elements.

4.1. Deformed distance fields

We use a penalty method for collision response. To compute the force a penetration depth value and a gradient at a collision point is required. This is a major problem for collisions especially in deformable objects. For each collision point it is difficult to find a minimal path to the surface and integrate

its length interactively because positions of the surface may change in each frame.

Deformed distance fields provide a simple solution for this problem. It creates distance fields inside the object and computes a penetration depth value and a gradient at an arbitrary point by interpolation. This process is performed as a pre-process. When a collision point is detected the depth values are interpolated.

4.2. Spatial hashing

For simple and efficient collision detection we use a spatial hashing method, which maps a discretized 3D space to a 1D indices. Firstly, we define a uniform grid on 3D space and a hash table by using this grid. All the tetrahedron mesh vertices are stored in this table. We next find neighboring vertices that are involved in other tetrahedral elements by tracing all the grid cells covered by such tetrahedrons. When vertices are stored in each cell, barycentric coordinates relative to this tetrahedron are computed for intersection tests.

A barycentric coordinate $\mathbf{b} = (b_0, b_1, b_2)^T$ of a vertex p at position $\mathbf{x}$ can be computed by

$$\begin{aligned} \mathbf{b} &= \mathbf{A}^{-1}(\mathbf{x} - \mathbf{x}_{t_0}), \\ \mathbf{A} &= [\mathbf{x}_{t_1} - \mathbf{x}_{t_0}, \mathbf{x}_{t_2} - \mathbf{x}_{t_0}, \mathbf{x}_{t_3} - \mathbf{x}_{t_0}], \end{aligned} \tag{1}$$

where $\mathbf{x}_{t_0} \ldots \mathbf{x}_{t_3}$ are four vertices positions of a tetrahedron. A vertex p lies in the tetrahedron if $b_0 \geq 0, b_1 \geq 0, b_2 \geq 0$ and $b_0 + b_1 + b_2 \leq 1$.

4.3. Reaction force

Penetration depth values and gradients at the collision points are computed by barycentric interpolation as described above. In this subsection we show the actual computation of reaction forces.

For all the tetrahedral mesh vertices we precompute the penetration values and gradients at their positions. Let's assume that the collision test finds a penetrating vertex p in the other tetrahedron t. A depth value and a gradient vector, d and $\mathbf{g}$, are computed by barycentric interpolation:

$$\begin{aligned} \mathbf{b}' &= (1 - b_0 - b_1 - b_2, b_0, b_1, b_2)^T, \\ d &= (d_{t_0}, d_{t_1}, d_{t_2}, d_{t_3})\mathbf{b}', \\ \mathbf{g} &= [\mathbf{g}_{t_0}, \mathbf{g}_{t_1}, \mathbf{g}_{t_2}, \mathbf{g}_{t_3}]\mathbf{b}', \end{aligned}$$

where $d_{t_0 \ldots t_3}$ are pre-computed penetration depth values at the tetrahedron vertices.

We should consider, however, that the gradient vectors $\mathbf{g}_{t_0 \ldots t_3}$ are followed to the current orientation of the tetrahedral element. We compute a rotation matrix from a pair of three vectors in a tetrahedron. One of a pair is vectors in the current state ($\mathbf{A}$ in Equation (1)), the other is in the initial state ($\mathbf{A}_0$ as calculated in (1)). A rotation matrix is defined as $\tilde{\mathbf{R}}$ which is an orthogonalized matrix of $\mathbf{R} = \mathbf{A}_0^{-1} \cdot \mathbf{A}$ (see details in [MDM*02]). We then rotate the gradient vector by using these matrices.

Let a precomputed gradient vector of a mesh vertex be $\mathbf{g}^0$. We rotate $\mathbf{g}^0$ by the rotation matrices of tetrahedrons $\mathbf{R}_{t_1} \ldots \mathbf{R}_{t_n}$ which include the vertex and a gradient vector $\mathbf{g}$ in the next step is defined by a normalized vector of the sum of their rotated vectors.

$$\begin{aligned} \mathbf{g}_{sum} &= \sum_{i=1}^{n} \mathbf{R}_{t_i} \mathbf{g}^0, \\ \mathbf{g} &= \frac{\mathbf{g}_{sum}}{|\mathbf{g}_{sum}|}. \end{aligned}$$

By this means $\mathbf{g}_{t_0 \ldots t_3}$ are recomputed from precomputed vectors $\mathbf{g}^0_{t_0 \ldots t_3}$. We finally compute reaction forces by a penalty method. A force vector $\mathbf{f}$ at a penetrating vertex p is expressed as follows:

$$\mathbf{f} = a \cdot d \cdot \mathbf{g},$$

where a denotes a user-defined penalty coefficient.

	Scene A	Scene B	Scene C
Tetrahedrons	6902	1812	883
Vertices	2532	792	358
Polys	132728	158792	19996
Fps	9-10	14-15	47-48

Table 1: *Statistical results of performance in our system*

5. Results and discussion

We test our algorithm for several scenes on a PC environment with Pentium 4 3.2GHz and GeForce FX 5950 (Table 1). Performance depends on the number of iterations of implicit integration, therefore we perform with a static number of iterations for evaluation relative to the complexity of meshes.

The FEM solution is not accurate whereas we can obtain enough speeds and realistic rendering for an interactive animation. Figure 2 shows four horses fall and collide with each other. In another example we represent a scene with throwing tori to characters. (Figure 3). From our experience several hundreds of tetrahedral elements for each object provide a proper balance in terms of computational costs and visual appearance. Our method can handle a scene which consists of up to several hundreds thousands of polygons.

6. Conclusion and future work

In this paper, we have proposed an interactive physically-based animation system for dense meshes. We have also provided a solution for collision response.

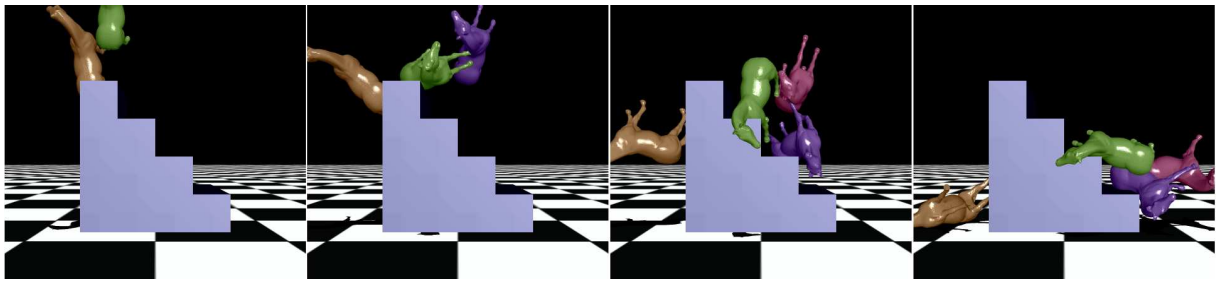

Figure 2: *Scene B: Falling horses*

One future direction is a skeleton-driven animation with collisions. While we use uniform elasticity models for deformations, most of creatures have bones and their motions are skeleton-based. In the same way most of physical interactions are firstly applied on muscles and then internal bones are moved according to the deformation of such muscles. It would help to create realistic character animations.

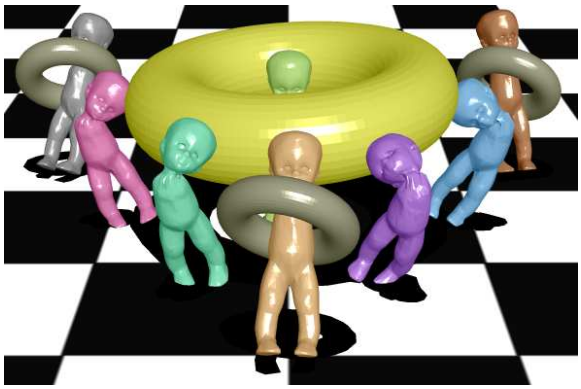

Figure 3: *Scene A: A scene with throwing tori to characters*

References

[Bat82] BATHE K.-J.: *Finite Element Procedures in Engineering*. Prentice-Hall, 1982.

[CGC*02] CAPELL S., GREEN S., CURLESS B., DUCHAMP T., POPOVIĆ Z.: A multiresolution framework for dynamic deformations. In *Proc. ACM SIGGRAPH Symposium on Computer Animation SCA 2002* (2002), ACM Press, New York, pp. 41–47.

[FL01] FISHER S., LIN M. C.: Deformed distance fields for simulation of non-penetrating flexible bodies. In *Computer Animation and Simulation 2001* (2001), Springer-Verlag, pp. 99–111.

[FvdPT97] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Dynamic free-form deformations for animation synthesis. In *IEEE Transactions on Visualization and Computer Graphics* (July 1997), vol. 3, IEEE Press, pp. 201–214.

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proc. SIGGRAPH 97* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 209–216.

[HSO03] HAUSER K. K., SHEN C., O'BRIEN J. F.: Interactive deformations using modal analysis with constraints. In *Proceedings of Graphics Interface 2003* (June 2003), A K Peters, pp. 247–256.

[JP99] JAMES D. L., PAI D. K.: Artdefo - accurate real time deformable objects. In *Proc. ACM SIGGRAPH: Computer Graphics* (1999), pp. 65–72.

[JP03] JAMES D. L., PAI D. K.: Multiresolution green's function methods for interactive simulation of large-scale elastostatic objects. In *ACM Transactions on Graphics* (2003), vol. 22, pp. 47–82.

[MDM*02] MÜLLER M., DORSEY J., MCMILLAN L., JAGNOW R., CUTLER B.: Stable real-time deformations. In *Proc. ACM SIGGRAPH Symposium on Computer Animation SCA 2002* (2002), ACM Press, New York, pp. 49–54.

[PW89] PENTLAND A., WILLIAMS J.: Good vibrations: Modal dynamics for graphics and animation. In *Proc. SIGGRAPH 89* (July 1989), ACM Press, New York, pp. 215–222.

[Sch97] SCHOBERL J.: Netgen - an advancing front 2D/3D-mesh generator based on abstract rules. In *Computations in Visualization and Science* (1997), vol. 1, pp. 41–52.

[THM*03] TESCHNER M., HEIDELBERGER B., MÜLLER M., POMERANETS D., GROSS M.: Optimized spatial hashing for collision detection of deformable objects. In *Proc. Vision, Modeling, Visualization VMV'03* (2003), Munich, Germany, pp. 47–54.

[TS86] T.W.SEDERBERG, S.R.PARRY: Free-form deformation of solid geometric models. In *Proc. ACM SIGGRAPH: Computer Graphics* (Aug 1986), vol. 25, ACM Press, New York, pp. 23–26.

[YSY01] YOSHITAKA C., SEIICHI K., YOSHIAKI O.: A particle method for elastic and visco-plastic structures and fluid-structure interactions. In *Computational Mechanics* (2001), vol. 27, pp. 97–106.