Volume Conserving Simulation of Deformable Bodies

Raphael Diziol Jan Bender Daniel Bayer

Universität Karlsruhe (TH), Karlsruhe, Germany

Abstract

We present a new method for simulating volume conserving deformable bodies using an impulse-based approach. In order to simulate a deformable body a tetrahedral model is generated from an arbitrary triangle mesh. All resulting tetrahedrons are assigned to volume constraints which ensure the conservation of the total volume. For the simulation of such a constraint impulses are computed and applied to the particles of the assigned tetrahedrons. The algorithm is easy to implement and ensures exact volume conservation in each simulation step.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Physically based modeling, I.3.7 [Computer Graphics]: Animation

1. Introduction

The dynamic simulation has become an important area of research in computer graphics and has many applications such as virtual environments, movie special effects and games. Topics like multi-body systems, water simulation and cloth simulation have been researched. The task of simulating deformable bodies is still a challenging problem, especially when the volume of the body has to be conserved.

In this paper a new method for simulating volume conserving deformable bodies using an impulse-based approach is presented. The impulse-based dynamic simulation describes a body as a set of particles linked by constraints. These constraints are satisfied by the computation of impulses. Constraints can be used to model joints, collisions and permanent contacts. In this paper the constraints enforce a body to conserve its total volume. Therefore a tetrahedral mesh is built to describe the volume. The tetrahedrons are assigned to constraints which are solved iteratively during simulation. The original triangle mesh can be assigned to the tetrahedral mesh and used for the visualization while the tetrahedral mesh is used for the simulation. With the impulsed-based approach even cyclic dependencies can be solved and no special treatment for the interaction with other objects like rigid bodies is needed.

2. Related Work

Since the first general physical model for the simulation of two- and three-dimensional deformable objects was presented [TPBF87], different approaches for simulating cloth and deformable bodies were studied. In [THMG04] a method for simulating deformable solids was introduced, where triangle and tetrahedral meshes with up to thousand primitives were simulated at interactive speed. The simulation could handle elastic and plastic deformations but could not guarantee the conservation of the total volume at each point of time. [MHTG05] showed mesh deformations where no connectivity is needed. They match a set of moved particles with the original ones by minimizing an energy function to restore the volume as good as possible. In the area of finite element simulation [ISF07] presented an approach adapted from fluid dynamics simulation, where the volume was conserved in a one-ring of tetrahedrons by making the velocities divergence free. The tetrahedral model was generated from a level set as described in [TMFB05]. The interaction between rigid and deformable bodies is presented in [RMSG*08]. More control of the deformations can be achieved by using key frame interpolation as presented in [AOW*08]. In this paper an impulse-based approach is used, because of its performance, stability and simplicity [Ben07].

3. Tetrahedral mesh

In order to represent the deformable bodies a volumetric structure must be created. Tetrahedral meshes are commonly used for this purpose. Spillmann et. al. [SWT06] presented an algorithm which creates tetrahedral meshes from arbitrary triangle soups. Thus, this approach can handle objects where the enclosed volume is not defined. We extend their algorithm for our purpose to generate the volumetric representation.

The key idea of the approach is to generate a pseudo volume from a triangle mesh. In order to determine the pseudo volume a distance field is computed in the first step. A standard approach as described in [Bær05] is used for this. The signs of the distances are determined by casting rays through the field. Let the distance field be defined in the axis-aligned bounding box of the mesh subdivided into voxels. The probability

$$P(\mathbf{x}) = 1 - c|d_{\min}(\mathbf{x})|$$

is assigned to each center $\mathbf{x}$ of a voxel, where $d_{\min}(\mathbf{x})$ is the distance from $\mathbf{x}$ to the closest surface point and c is a normalization parameter so that $0 \leq P(\mathbf{x}) \leq 1$. Each voxel must be classified whether it is part of the pseudo volume or not. The classification is made by casting rays from different directions through the distance field. For each voxel traversed by a ray the probability $P(\mathbf{x})$ is used to make the decision. The final sign of a voxel is determined by a majority vote from all rays.

When casting a ray with direction $\mathbf{v}$ through the distance field the probability $P(\mathbf{x})$ is used to check how likely it is that the ray enters or leaves the pseudo volume. Two user defined thresholds P_{in} and P_{subs} are used to make the decision. The ray enters or leaves the volume if $P(\mathbf{x}) > P_{\text{in}}$ and $\frac{\partial P}{\partial \mathbf{v}}(\mathbf{x}) > 0$. A subsequent volume change is only reported if $P(\mathbf{x}) < P_{\text{subs}}$ and $\frac{\partial P}{\partial \mathbf{v}}(\mathbf{x}) < 0$ in the meantime. This gives control to the user to finely adjust the resulting pseudo volume, especially for thin objects as shown in Figure 1. An adequate approximation of the original triangle mesh is sufficient for our purpose, because the triangles of the original mesh can be used for the visualization using freeform deformations as described in [MJBF02].

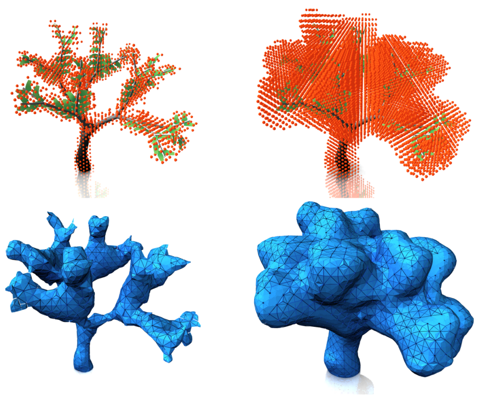
Figure 1: *A tree with its different pseudo volumes depending on the user defined thresholds (top) and the resulting tetrahedral models (bottom).*

After each voxel is classified whether it is part of the pseudo volume or not, an uniform lattice with cells is generated onto the distance field, where each cell belonging to the pseudo volume is subdivided into five tetrahedrons. A cell with volume V lies in the pseudo volume if the pseudo density

$$\rho(V) = \frac{m(V)}{M(V)}$$

is larger than a user defined threshold. $M(V)$ is the total number of voxels within V and $m(V)$ is the number of voxels within the pseudo volume. To avoid small holes at the boundary where no tetrahedrons have yet been generated, cells with more than three adjacent cells already belonging to the pseudo volume are iteratively added. After the tetrahedrons are generated, sharp corners must be smoothed. Thus, a second order laplacian filter [DMSB99] with volume preservation is applied to the surface of the tetrahedral mesh. More symmetrical and smoother results are obtained when only considering the vertices in the umbrella operator which are connected over axis-aligned edges in the unsmoothed tetrahedral mesh. If vertices connected by diagonal edges are also considered in the umbrella operator, then sharp corners are smoothed depending on the triangulation of the surface. Symmetrical smoothed models are important for the simulation in order to guarantee symmetrical deformations. See Figure 2 for the differences between the standard smoothing and our variant. After the computation of the distance field the user can interactively change the resolution of the lattice, the thresholds and smoothing parameters to adjust the model to his needs. The tetrahedral mesh can be adapted to a coarse mesh like the convex hull as well to a finer detailed mesh by adjusting the parameters of the algorithm.

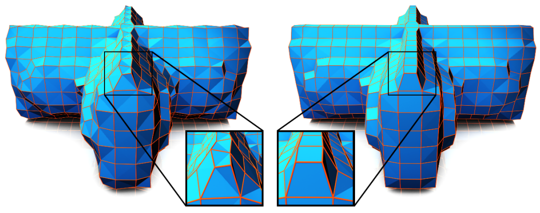
Figure 2: *An object after two smoothing iterations. Considering vertices connected by diagonal edges in the umbrella operator (left) results in a less symmetric smoothing compared to our variant (right).*

4. Volume constraint

After the tetrahedral mesh has been computed each vertex of the mesh gets assigned a particle. The dynamic state of a particle is defined by its mass m, its position $\mathbf{p}(t)$ and its velocity $\mathbf{v}(t)$. Each cell consisting of five tetrahedrons gets assigned a volume constraint. The volume constraint consists of a volume preserving part and 16 springs which are introduced as external forces. On each edge of the cell and over the diagonals springs are placed as depicted in Figure 3. The springs force the cell to preserve its original form while the volume is recovered during a simulation step. Forcing only one tetrahedron to conserve its volume could result in locking as stated in [ISF07]. Using five tetrahedrons at once gives the model enough freedom for deforming.

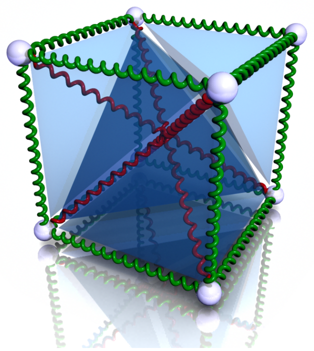

Figure 3: *The volume constraint with its five tetrahedrons, eight particles, and* 16 *springs. The red springs can have different spring constants from the green ones.*

In each simulation step with time step size h the positions of the particles are integrated according to

$$\begin{aligned}\mathbf{v}_i(t_0+h) &= \mathbf{v}_i(t_0)+\int_{t_0}^{t_0+h}\frac{\mathbf{F}_{\text{ext}}}{m_i}dt \\ \mathbf{p}_i(t_0+h) &= \mathbf{p}_i(t_0)+\int_{t_0}^{t_0+h}\mathbf{v}(t)dt \quad =: \mathbf{x}_i \qquad (1)\end{aligned}$$

with external forces $\mathbf{F}_{\text{ext}}$ and masses m_i. In order to conserve the initial volume V_0 of a cell the volume $V(t+h)$ must satisfy the constraint

$$V(t+h)-V_0=0 \ . \qquad (2)$$

Thus, impulses for the eight particles of the cell are computed and applied. These impulses are determined by using a preview of the constraint state. The integrated particle positions $\mathbf{x}_j, \mathbf{x}_k, \mathbf{x}_l$ and $\mathbf{x}_m$ from Equation 1 are used in a first step to compute the integrated volumes

$$V_i(t+h)=\frac{1}{6}\left(\mathbf{x}_j-\mathbf{x}_m\right)\left((\mathbf{x}_k-\mathbf{x}_m)\times(\mathbf{x}_l-\mathbf{x}_m)\right) \qquad (3)$$

of each tetrahedron. The volume $V(t+h)$ of a cell is computed as the sum $\sum_{i=1}^{5}V_i(t+h)$ of the five tetrahedron volumes. Due to the fact, that the integrated volume does not generally satisfy Equation 2, the particles have to be moved in a second step. Moving the particles by $\lambda(\mathbf{x}_i-\mathbf{c})$ the center of mass $\mathbf{c}=\frac{1}{8}\sum_{i=1}^{8}\mathbf{x}_i$ is conserved while restoring the initial volume of the cell. Exploiting the fact, that the resulting volume of a tetrahedron is $\lambda^3 V_i(t+h)$ after the particles have been moved, the volume of a cell thus is $\lambda^3\sum_{i=1}^{5}V_i(t+h)$. With Equation 2 it can be seen that choosing $\lambda=\sqrt[3]{\frac{V_0}{V(t+h)}}$ restores the initial volume. Thus, the needed impulses for the particles are

$$\Delta\mathbf{v}_i=\frac{\lambda-1}{hm_i}(\mathbf{x}_i-\mathbf{c}) \ , \qquad (4)$$

where m_i is the mass and $\Delta\mathbf{v}_i$ is the velocity change of particle i. According to Newton's second law of motion the conservation of momentum has to be guaranteed. Equation 4 holds this property if the masses of each particle are equal, because

$$\sum_{i=1}^{8}\frac{\lambda-1}{hm}(\mathbf{x}_i-\mathbf{c}) \ = \ \frac{\lambda-1}{hm}\sum_{i=1}^{8}\left(\mathbf{x}_i-\frac{1}{8}\sum_{j=1}^{8}\mathbf{x_j}\right)=\mathbf{0} \ .$$

We check the sign of Equation 3 while enforcing the volume constraint to avoid inverting a tetrahedron. If the sign of the volume of one tetrahedron is negative, we simply halve the time step size h as long as no volume is negative and then compute the constraint with the new time step size. In general this problem only occurs, if a tetrahedron is not well shaped or very strong external forces are involved.

The impulse-based simulation corrects the constraints iteratively until all constraints are satisfied. This process converges to the physical correct solution as shown in [SBP05]. Collision handling was also integrated using the impulse-based method as described in [BS06].

5. Results

The presented tetrahedral model generation and volume constraint was implemented in C++. In each simulation step the volume of the deformable body is conserved. The impulse-based dynamic simulation also allows interaction between deformable bodies and rigid bodies. Figure 4 shows two animations where a rigid sphere collides with a deformable cube. In the first case we have used strong springs to disallow strong deformations of the body. The second case shows the same animation where the diagonal spring constants of the cells were set to a low value. Thus, a more flexible deformation was possible.

6. Conclusion

In this paper a new method for simulating deformable bodies with an impulsed-based approach was presented. The tetrahedral model generation algorithm produces volumetric models even when no enclosed volume is defined. The parameters of the algorithm let the user interactively define different resolutions and appearances of the resulting model. Due to the special smoothing we obtain symmetrical models which are important for symmetric deformations. The tetrahedral model is used for the simulation while the original triangle mesh can be used for the visualization. Taking care in the smoothing process, that the resulting tetrahedrons do not get too small and are well shaped, makes the inversion of a tetrahedron unlikely, resulting in shorter simulation times due to bigger adaptive time step sizes. The springs define the stiffness of the deformable bodies resulting in different behaviors of the deformations.

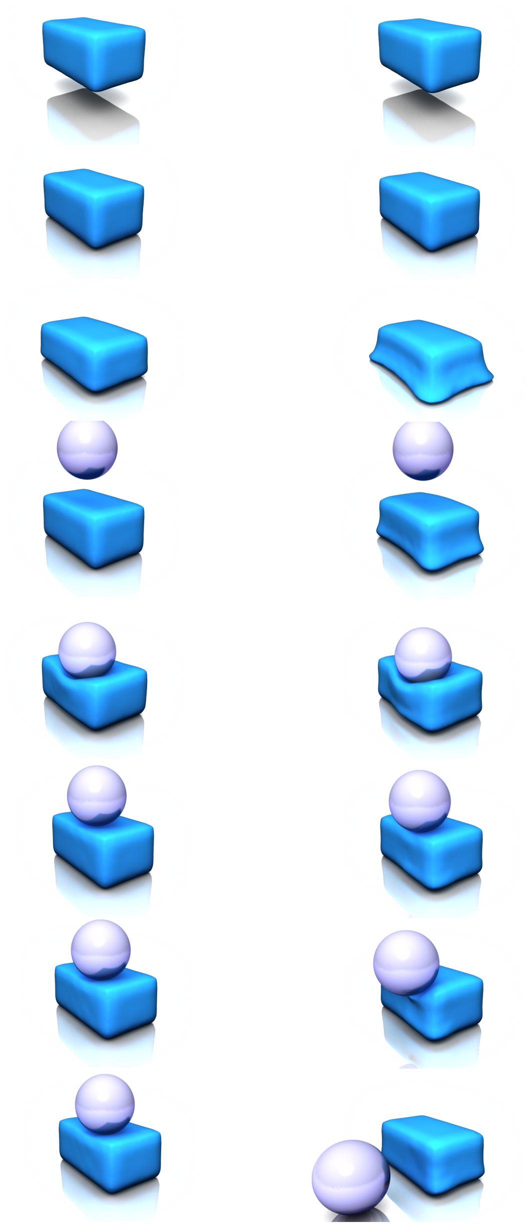

Figure 4: *Two animations of a deformable body (top to bottom): The diagonal springs on the right side are weaker resulting in a more flexible deformation.*

References

[AOW*08] Adams B., Ovsjanikov M., Wand M., Seidel H.-P., Guibas L. J.: Meshless modeling of deformable shapes and their motion. In *ACM SIGGRAPH/Eurographics SCA* (2008).

[Bær05] Bærentzen A.: Robust generation of signed distance fields from triangle meshes. In *Fourth International Workshop on Vol. Graphics* (2005), pp. 167–175.

[Ben07] Bender J.: Impulse-based dynamic simulation in linear time. *Computer Animation and Virtual Worlds* (2007), 225–233.

[BS06] Bender J., Schmitt A.: Constraint-based collision and contact handling using impulses. In *Proceedings of the 19th international conference on computer animation and social agents* (2006), pp. 3–11.

[DMSB99] Desbrun M., Meyer M., Schröder P., Barr A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proc. SIGGRAPH* (1999), pp. 317–324.

[ISF07] Irving G., Schroeder C., Fedkiw R.: Volume conserving finite element simulations of deformable models. In *Proc. SIGGRAPH* (2007), pp. 13:1–13:6.

[MHTG05] Müller M., Heidelberger B., Teschner M., Gross M.: Meshless deformations based on shape matching. *ACM Trans. Graph.* (2005).

[MJBF02] Milliron T., Jensen R. J., Barzel R., Finkelstein A.: A framework for geometric warps and deformations. *ACM Trans. Graph.* (2002), 20–51.

[RMSG*08] Robinson-Mosher A., Shinar T., Gretarsson J., Su J., Fedkiw R.: Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. Graph.* (2008), 1–9.

[SBP05] Schmitt A., Bender J., Prautzsch H.: *On the Convergence and Correctness of Impulse-Based Dynamic Simulation.* Internal Report 17, Universität Karlsruhe, 2005.

[SWT06] Spillmann J., Wagner M., Teschner M.: Robust tetrahedral meshing of triangle soups. In *Proc. Vision, Modeling, Visualization VMV'06* (2006), pp. 9–16.

[THMG04] Teschner M., Heidelberger B., Müller M., Gross M.: A versatile and robust model for geometrically complex deformable solids. In *Proceedings of Computer Graphics International* (2004), pp. 312–319.

[TMFB05] Teran J., Molino N., Fedkiw R., Bridson R.: Adaptive physics based tetrahedral mesh generation using level sets. *Eng. with Comput. 21*, 2-18 (2005).

[TPBF87] Terzopoulos D., Platt J., Barr A., Fleischer K.: Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (1987), pp. 205–214.