A Versatile and Robust Model for Geometrically Complex Deformable Solids

Matthias Teschner Bruno Heidelberger Matthias Müller Markus Gross

Computer Graphics Laboratory
ETH Zurich

Abstract

In this paper, we present a versatile and robust model for geometrically complex deformable solids. Our approach can be applied to deformable tetrahedral meshes and to deformable triangle meshes. The model considers elastic and plastic deformation. It handles a large variety of material properties ranging from stiff to fluid-like behavior. Due to the computational efficiency of our approach, complex environments consisting of up to several thousand primitives can be simulated at interactive speed.

The presented approach to deformable modeling is part of a simulation environment with integrated collision handling for tetrahedral meshes. For visualization purposes, tetrahedral meshes can be coupled with high-resolution surface meshes. Results are presented for deformable tetrahedral meshes and for deformable triangle meshes which are used to represent cloth and discrete shells.

Key words: Physically-based Modeling, Computer Animation, Deformable Modeling, Collision Handling, Cloth Simulation, Discrete Shells

1 Introduction

There is a growing demand for interactive deformable modeling in computational surgery and entertainment technologies, especially in games and movie special effects. These applications do not necessarily require physically-correct deformable models, but efficient deformable models with physically-plausible dynamic behavior. Additionally, simulations should be robust and controllable, and they should run at interactive speed.

This paper describes a unified method suitable for modeling deformable tetrahedral or triangulated meshes with elasticity and plasticity. The proposed model extends existing deformable modeling techniques by incorporating efficient ways for volume and surface area preservation. The computational efficiency of our approach is similar to simple mass-spring systems. Thus, environments of up to several thousand deforming primitives can be handled at interactive speed.

In order to optimize the dynamics computation various numerical integration schemes have been compared. Comparisons have been performed with respect to robustness and performance in the context of our model.

The proposed deformable modeling approach is part of a simulation environment. This environment can handle collisions between tetrahedral meshes. For visualization purposes, deformable tetrahedral meshes can be coupled with high-resolution surface meshes in the spirit of FFD.

The paper presents experiments with various scenes of dynamically deforming tetrahedral meshes. Further, applications of the deformable model to interactive cloth simulation and interactive discrete shells are presented.

2 Related Work

Deformable models have been extensively investigated in the last two decades [24, 25, 1, 12]. Approaches based on mass-spring models [4], particle systems [9], or FEM [17, 20] have been used to efficiently represent 3D objects or deformable 2D structures, such as cloth [29] or discrete shells [14]. Typical applications for these approaches can be found in computational surgery [8] and games. Although very efficient algorithms have been proposed [7, 8, 30], only a few hundred deformable primitives have been simulated in real-time to date.

While many approaches are restricted to elastic deformation, models for plastic deformation have been introduced in [26, 21] . However, no real-time approximations of these models have been presented so far.

In [14], a method to simulate the dynamic behavior of discrete shells has been described. Very promising results have been shown. However, the approach is computationally expensive.

Many approaches focus on solutions to specific problems in deformable modeling. However, there exist no efficient, unified approach to physically-plausible simulation of 2D and 3D deformable models with elasticity and plasticity. Further, there exist no framework where complex deformable objects can be handled with integrated collision handling at interactive rates.

3 Deformable Model

We consider deformable solids that are discretized into tetrahedra and mass points. In order to compute the dynamic behavior of objects we derive forces at mass points from potential energies (see Sec. 3.1). These forces preserve distances between mass points (see Sec. 3.2), they preserve the surface area of the object (see Sec. 3.3), and they preserve the volume of tetrahedra (see Sec. 3.4). The material properties of a deformable object are described by weighted stiffness coefficients of all considered potential energies.

3.1 Potential Energies

In order to represent deformations of objects, we consider constraints of the form $C(\mathbf{p}_0, \ldots, \mathbf{p}_{n-1})$. These scalar functions depend on mass point positions $\mathbf{p}_i$. They are zero if the object is undeformed. In order to compute forces based on these constraints we consider the potential energy

$$E(\mathbf{p}_0, \ldots, \mathbf{p}_{n-1}) = \frac{1}{2} k C^2 \tag{1}$$

with k denoting a stiffness coefficient. This coefficient has to be defined for each type of potential energy. The potential energy is zero if the object is undeformed. Otherwise, the energy is larger than zero. The potential energies of our model are independent of rigid body modes of the object. The overall potential energy derived from our constraints can be interpreted as deformation energy of the object. Now, forces at mass points $\mathbf{p}_i$ are derived as

$$\mathbf{F}^i(\mathbf{p}_0, \ldots, \mathbf{p}_{n-1}) = -\frac{\partial}{\partial \mathbf{p}_i} E = -kC \frac{\partial C}{\partial \mathbf{p}_i} \tag{2}$$

The overall force at a mass point is given as the sum of all forces based on potential energies that consider this mass point. Damping which significantly improves the robustness of the dynamic simulation can be incorporated as

$$\begin{aligned} &\mathbf{F}^i(\mathbf{p}_0, \ldots, \mathbf{p}_{n-1}, \mathbf{v}_0, \ldots, \mathbf{v}_{n-1}) \\ &= \left(-kC - k_d \sum_{0 \le j < n} \frac{\partial C}{\partial \mathbf{p}_j} \mathbf{v}_j \right) \frac{\partial C}{\partial \mathbf{p}_i} \end{aligned} \tag{3}$$

with $\mathbf{v}_i$ denoting the velocity of a mass point and k_d denoting the damping coefficient. We do not consider any additional constraints or boundary conditions for our forces. In contrast to similar approaches [4, 5, 22], we do not explicitly bound potential energies or forces resulting from the energies.

The direction of a force $\mathbf{F}$ based on a potential energy E corresponds to the negative gradient of E, i. e., a dynamic simulation resulting from these forces reduces the deformation energy of an object. Further, these forces are orthogonal to rigid body modes, i. e. they conserve linear and angular momentum of the object.

3.2 Distance Preservation

The first potential energy E_D considers all pairs of mass points that are connected by tetrahedral edges. E_D represents energy based on the difference of the current distance of two points and the initial or rest distance D_0 with $D_0 \neq 0$:

$$E_D(\mathbf{p}_i, \mathbf{p}_j) = \frac{1}{2} k_D \left(\frac{|\mathbf{p}_j - \mathbf{p}_i| - D_0}{D_0} \right)^2 \tag{4}$$

Forces $\mathbf{F}_D$ resulting from this energy are computed as stated in (3). While damping of theses forces is very useful to improve the stability of the numerical integration process, experiments have shown no significant improvement of the stability if damping is applied to forces resulting from the other two energies that we consider in our deformation model.

3.3 Surface Area Preservation

The second energy E_A considers triples of mass points that build surface triangles. E_A represents energy based on the difference of the current area of a surface triangle and its initial area A_0 with $A_0 \neq 0$:

$$\begin{aligned} &E_A(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k) \\ &= \frac{1}{2} k_A \left(\frac{\frac{1}{2} |(\mathbf{p}_j - \mathbf{p}_i) \times (\mathbf{p}_k - \mathbf{p}_i)| - A_0}{A_0} \right)^2 \end{aligned} \tag{5}$$

Forces $\mathbf{F}_A$ based on this energy are computed as stated in (2). As already mentioned in Sec. 3.2 these forces are not damped in our approach. Preservation of surface area is considered in the animation of discrete shells and thin plates. In the animation of tetrahedral meshes as shown in Sec. 6.1, 6.2, and 6.3 the effect of this energy is negligible.

3.4 Volume Preservation

Our third potential energy E_V considers sets of four mass points that build tetrahedra. E_V represents energy based on the difference of the current volume of a tetrahedron and its initial volume V_0:

$$\begin{aligned} &E_V(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k, \mathbf{p}_l) \\ &= \frac{k_V}{2} \frac{\left(\frac{1}{6} (\mathbf{p}_j - \mathbf{p}_i) \cdot ((\mathbf{p}_k - \mathbf{p}_i) \times (\mathbf{p}_l - \mathbf{p}_i)) - V_0 \right)^2}{\tilde{V}_0^2} \end{aligned} \tag{6}$$

with $\tilde{V}_0 = V_0$ if our model is applied to volumetric tetrahedral meshes as presented in Sec. 6.1. In this case we assume $V_0 \neq 0$. However, if our model is applied to thin plates or discrete shells, we can not assume $V_0 \neq 0$. In this case we use $\tilde{V}_0 = \frac{\sqrt{2}}{12} \bar{l}$ with $\bar{l}$ denoting the average edge length of a tetrahedron. Then, $\tilde{V}_0$ corresponds to the volume of a regular tetrahedron with edge length $\bar{l}$. Based on E_V forces $\mathbf{F}_V$are computed as stated in (2).

The preservation of the signed volume as it is calculated with the mixed product in (6) is of major importance to our deformation model. In contrast to the energies E_D and E_A, which are not sensitive to inverted tetrahedra, forces based on E_V preserve the initial orientation of the vectors in the mixed product. If a tetrahedron is inverted and the orientation of these vectors changes, the sign of the volume represented with the mixed product in (6) changes accordingly. Thus, inverting a tetrahedron results in forces $\mathbf{F}_V$ that restore the original orientation of the tetrahedron.

Due to the normalization of all constraints that are considered in the potential energies the stiffness coefficients k_D, k_A, k_V are scale-invariant. These stiffness coefficients can be used to mimic a wide range of material properties as presented in Sec. 6. Refer to Sec. 6.4 for an overview of sets of stiffness coefficients for various materials.

Fig. 1 illustrates the three types of forces employed in our deformation model.

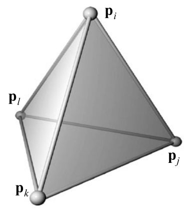

Figure 1: A tetrahedron with four mass points is the basic volumetric primitive of our deformable model. In this simple example, six distance-preserving forces between all pairs of points, e. g. $F_D(\mathbf{p}_i, \mathbf{p}_j)$, four area-preserving forces, e. g. $F_A(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$, and the volume-preserving force $F_V(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k, \mathbf{p}_l)$ are considered.

4 Tetrahedral and Triangulated Meshes

The proposed deformable model is designed to work with tetrahedral meshes. However, the proposed deformation model can also be applied to arbitrary triangle meshes. In this case, distance-preserving forces are considered for all edges and area-preserving forces are considered for all triangles of the mesh. Employing these forces, a dynamic simulation preserves all distances between mass points and the surface area. However, there is no resistance of the model against bending which is essential in cloth and discrete shell simulation [2, 4, 10, 15, 22, 14].

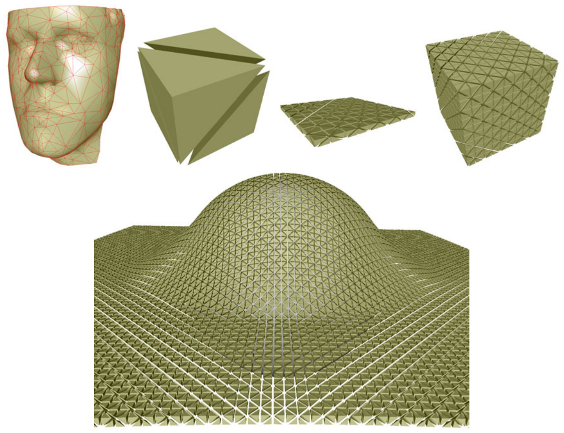

Figure 2: Models used for performance measurements: cubes, cuboid, face, membrane. Geometrical complexity and computing times are given in Tab. 1.

In order to control bending of triangulated surface we generate a tetrahedron for each pair of adjacent triangles with one common edge. If vertices opposite to this common edge are connected with an additional edge, a virtual tetrahedron is generated.

Now, preservation of the volume of the virtual tetrahedron, while also preserving the area of the triangles, corresponds to a preservation of the angle between the two adjacent triangles. If this angle is zero, the volume of the virtual tetrahedron is zero. Otherwise, the volume is larger or smaller than zero depending on a concave or convex angle.

If the volume-preserving force is considered for a virtual tetrahedron, its volume and the respective angle between adjacent triangles are preserved. In case of thin plates or cloth, the rest angle is zero. In case of discrete shells, arbitrary rest angles can occur.

5 Numerical Integration

In order to compute the dynamic behavior of our deformable models, Newton's equation of motion is applied to all mass points. Based on initial values for positions and velocities, a time step h, internal forces at mass points resulting from our deformation energies, and external forces, such as gravity, piecewise linear trajectories for all mass points are calculated employing a numerical integration scheme.

Based on performance comparisons presented in Sec. 5.2 we have chosen the Verlet scheme for numerical integration [28]. This method has been very popular in molecular dynamics for decades and has recently been proposed in the context of physically-based simulation of cloth, general mass-spring systems, and rigid bodies [15, 10, 18]. The Verlet algorithm uses positions and forces at time t, and positions at the previous time $t - h$ to calculate new positions at time $t + h$:

$$\begin{aligned}\mathbf{x}(t+h) &= 2\mathbf{x}(t) - \mathbf{x}(t-h) + h^2\frac{\mathbf{F}(t)}{m} + O(h^4) \\ \mathbf{v}(t+h) &= \frac{\mathbf{x}(t+h) - \mathbf{x}(t-h)}{2h} + O(h^2) \qquad (7)\end{aligned}$$

with $\mathbf{F}(t) = \mathbf{F}_D(t) + \mathbf{F}_A(t) + \mathbf{F}_V(t)$. The Verlet method has several advantages in environments with interacting, dynamically deforming objects. First, only one force computation is required per integration step. This is essential, since force computation is the most expensive part in the calculation of an integration step. Second, the integration of positions has a local discretization error of $O(h^4)$. This high accuracy allows for comparatively large time steps. Third, the integration of positions is independent of the integration of velocities if undamped forces are used. Depending on the application this could be employed to omit the integration of velocities, which would further improve the performance. However, we do not use this property, since damping of distance-preserving forces is essential for the robustness of our model. Further, collision response, as utilized in some of the experiments shown in Sec. 6.1, requires velocities of mass points.

5.1 Computing Time

In our experiments, we distinguish between computational complexity of a numerical integration scheme, which is discussed in this section, and the performance of an integration method, which is discussed in the following section.

In a first experiment, we have tested the computational complexity of the Verlet scheme in environments with dynamically deforming objects of varying geometrical complexity. Tab. 1 shows the computing time for one numerical integration step with various deformable objects that are depicted in Fig. 2. Our measurements show that 1500 forces can be updated at 1 KHz, while more complex objects with 14000 forces can be updated at 140 Hz. Since we are interested in an interactive behavior of our simulations, it is essential to have update rates of the numerical integration, that are above 20 Hz.

setup	points	n_D	n_A	n_V	time [ms]
cube 1	8	18	0	5	0.03
cuboid	242	981	0	500	1.00
face	472	2622	874	1277	2.26
cloth	1301	5478	1826	2739	5.03
cube 2	1331	6930	0	5000	7.11
santa	915	7500	2500	3700	7.36
membrane	20402	90801	0	50000	114.61

Table 1: Computing time for one Verlet step (Intel Pentium 4, 2.8 GHz). The number of mass points is given for each model. Further, n_D, n_A, n_V denote the numbers of considered distance-preserving, area-preserving, and volume-preserving forces per integration step, respectively. In case of $n_A = 0$, the object surface is not considered in the deformation model (see Sec. 3.3).

Note, that the computing time for an integration step does not correspond to the performance of an integration method. In order to assess the performance, the ratio of integration time step and computing time has to be considered. This problem is addressed in the following section.

5.2 Comparison to Other Approaches

In order to optimize the performance of our dynamically deforming objects, we have implemented and compared several numerical integration methods that have been proposed in previous approaches to physically-based deformation of mass-spring or particle systems.

We are not only interested in maximal time steps or minimal computing time, but in optimal performance. Therefore, we propose to consider the ratio of the numerical integration time step and the computing time for one numerical integration time step as performance measure.

As a test case, we have applied various integration schemes to a cube with 1331 mass points, considering 5000 volume-preserving forces and 6930 distance-preserving forces. The cube falls onto a plane where collisions are handled (see Fig. 3). We have implemented and compared the following integration schemes: Verlet [28], velocity Verlet [23], Runge-Kutta, Beeman [3], explicit Euler, Leap-Frog [16], Heun, implicit Theta-Scheme, and Gear [11]. Tab. 2 shows measurements of time steps and computing times for various integration methods.

method	time step [ms]	comp. time [ms]	ratio
Verlet	3.1	7.3	0.427
Leap-frog	3.1	7.3	0.426
RK 2	4.9	14.3	0.342
vel. Verlet	2.5	7.3	0.341
Beeman	2.5	7.4	0.337
Heun	4.2	18.4	0.229
expl. Euler	1.5	7.3	0.205
RK 4	6.3	33.0	0.191

Table 2: Maximum time step and computing time for various integration schemes. The last column shows the ratio of time step and computing time for one integration step. This ratio can be interpreted as performance measure of an integration method.

Our measurements suggest, that Verlet and Leap-frog can be computed very efficiently, while providing a reasonable time step. Although the fourth order Runge-Kutta scheme allows for a larger time step, its computation is significantly more expensive.

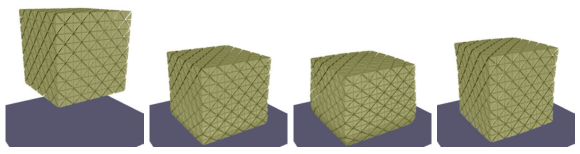
Figure 3: Image sequence illustrating our test scenario for the numerical integration schemes. Computing times and time steps are given in Tab. 2.

Two classes of integration schemes, namely predictor-corrector methods (Gear) and implicit methods (Theta), have been implemented and considered in the comparison, but are not listed in Tab. 2. Both methods suffer from drawbacks that are related to our deformable model and to our application.

Although the Gear scheme is very efficient and robust, we do not consider it in our environment. This is due to the fact, that the Gear scheme has to be re-initialized after collision handling which significantly reduces the stability of the method. Since collision handling is an important component in our environment, Gear is not appropriate for our application.

Implicit integration schemes have shown to be very robust in physically-based simulations [2, 7, 8, 13, 29]. They are very popular, since they allow for large time steps.

On the other hand, they are expensive to compute. Implicit methods require to solve a sparse linear system per integration step. Further, computing and storing the system matrix cause additional costs. In our application with comparatively complex objects with several thousand degrees of freedom, these costs are significant. Although, we use an efficient Conjugate Gradient algorithm with only a few iterations (5-30), we do not achieve computing times faster than 200 ms in our test scenario. This is not appropriate for our application.

A second aspect is the combination of dynamically deforming objects and collision handling. Larger time steps cause larger penetration depths of objects which are difficult to resolve. Robust collision response commonly requires a small intersection volume of two colliding objects. This is difficult to guarantee, if the time step is too large. In some of the experiments in Sec. 6.1, collision handling is employed. In these experiments, the limiting factor for the time step is not the numerical integration, but our collision handling scheme.

From our perspective, the optimal numerical integration scheme does not depend on the size of the time step. Instead, it depends on the underlying model, on the application, and on the complexity of the data structures. Although implicit integration methods are very useful in many applications, we propose to use Verlet or Leap-Frog for our specific problem.

6 Results

In this section, we describe some applications of our deformable model. Sec. 6.1 presents some examples with elastically deforming volumetric tetrahedral meshes. Sec. 6.1 also shows, how our approach can be applied to interactive simulations of cloth and discrete shells. Sec. 6.2 explains the incorporation of plasticity into our deformable model. Sec. 6.3 shows, how the variation of stiffness constants can be used to animate melting or fluid-like objects. Sec. 6.4 gives an overview of all parameters and the performance of the simulations.

6.1 Elastic Deformation

We have integrated our model into a simulation environment for deformable objects. In this environment, our tetrahedral meshes can be coupled with high-resolution surface meshes in the spirit of FFD [6, 19]. Fig. 4 illustrates this visualization technique. Further, collisions between deformable objects can be handled based on a spatial hashing approach presented in [27]. If more than one deformable tetrahedral mesh is used, this collision handling scheme is employed.

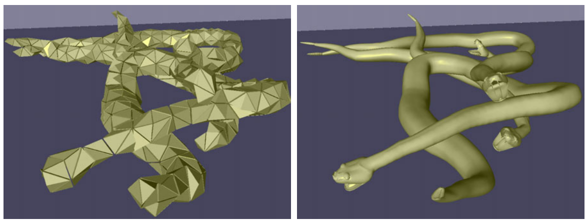

Figure 4: A low-resolution tetrahedral mesh and a high-resolution surface mesh of a snake. Deformation is computed for the tetrahedral mesh, while the high-resolution mesh is visualized. Refer to Sec. 6.4 for parameters and performance.

Fig. 5 and Fig. 6 show sequences of a deforming pitbull and deforming cows with collision handling. In addition to forces resulting from our collision response scheme, the simulation environment considers external forces, such as gravity and user-defined dragging of mass points.

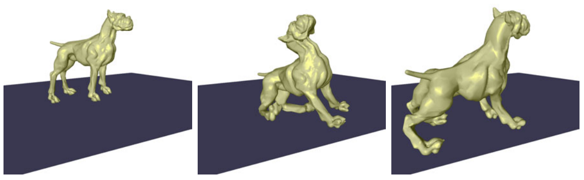

Figure 5: Sequence of a dynamically deforming pitbull. Refer to Sec. 6.4 for parameters and performance.

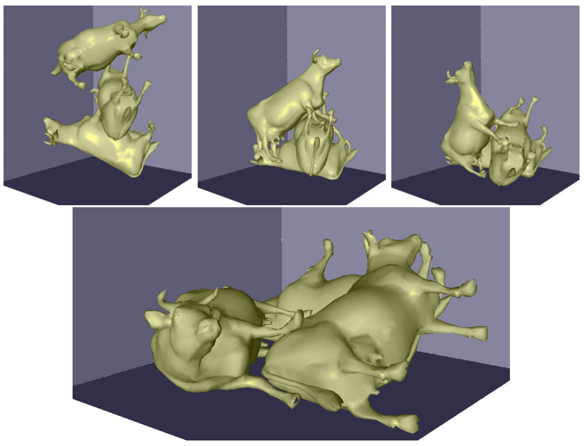

Figure 6: Sequence of dynamically deforming and colliding cows. Refer to Sec. 6.4 for parameters and performance.

Fig. 7 shows a sequence of interactive cloth simulation. Fig. 8 depicts a dynamically deforming face which is modeled as a discrete shell. These figures illustrate that our approach can also be employed to interactively animate thin plates and discrete shells.

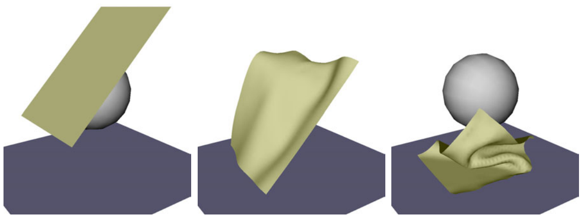

Figure 7: Sequence of a deformable cloth model interacting with a rigid sphere. Refer to Sec. 6.4 for parameters and performance.

6.2 Plastic Deformation

In addition to elastic deformation, the proposed model can also handle plastic deformation as introduced in [21,

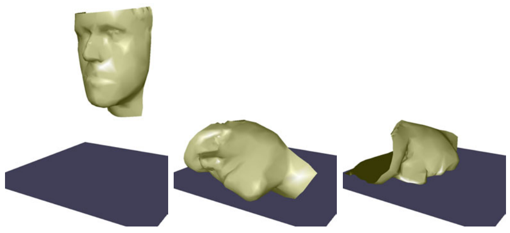

Figure 8: Sequence of a deformable face which is modeled as a discrete shell. Refer to Sec. 6.4 for parameters and performance.

26]. Therefore, the deformation of an object is decomposed into elastic and plastic deformation, whereas the plastic deformation does not contribute to the deformation energy of an object. In our model, this can be represented employing the energy E_D which represents distance differences (see Sec. 3.2). Therefore, E_D is decomposed into two components for elasticity and plasticity:

$$E_D = E_D^{elastic} + E_D^{plastic} \tag{8}$$

This corresponds to a decomposition of forces $\mathbf{F}_D$ resulting from E_D:

$$\mathbf{F}_D = \mathbf{F}_D^{elastic} + \mathbf{F}_D^{plastic} \tag{9}$$

In the case of elastic deformation, E_D contributes to the deformation energy of an object and F_D is applied to minimize this energy. However, if plasticity is considered, only the elastic part $E_D^{elastic}$ contributes to the deformation energy. Therefore, only $\mathbf{F}_D^{elastic} = \mathbf{F}_D - \mathbf{F}_D^{plastic}$ is considered accordingly in the numerical integration process (see Sec. 5).

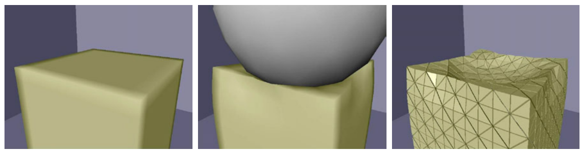

Figure 9: Sequence of a plastically deformed cube. Some deformation energy caused by the rigid sphere is stored. The right image shows the static equilibrium of the plastically deformed cube. Refer to Sec. 6.4 for parameters and performance.

In order to model the plastic evolution, O'Brien [21] proposes three parameters that we have implemented in our model. The first parameter specifies a minimum value for E_D that must be met before the decomposition of elasticity and plasticity occurs. This represents the fact, that small deformations are only elastic. The second parameter provides a maximum value for $E_D^{plastic}$. This parameter controls the maximum amount of deformation that can be stored by an object. The third parameter specifies the rate of plastic flow. This parameter can be used to model hysteresis of a material. Further details on these three parameters can be found in [21]. Fig. 9 illustrates the process of plastic deformation. Plastic deformation is only considered in the distance-preserving forces. Volume preservation and - if applied - surface area preservation is not affected by plasticity.

6.3 Melting

The proposed deformation approach allows to mimic a wide range of material properties. Obviously, hard and soft materials can be represented by adjusting the stiffness coefficient k_D for distance-preserving forces. Additionally, volume and area-preserving forces can be employed to model further properties.

Fig. 10 illustrates the versatility of our deformable model. In this example, the stiffness coefficient k_V for volume-preserving forces is significantly larger than the coefficient k_D for distance-preserving forces. This allows to animate melting objects, i. e. distances between adjacent mass points can vary heavily, while most deformation energy is used to preserve the volume of the object.

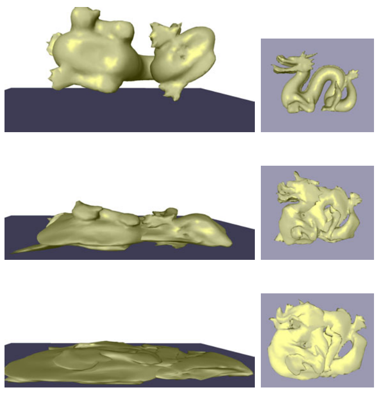

Figure 10: Sequence of a melting dragon. Pairs of left and right images illustrate the same simulation step from two view points. Refer to Sec. 6.4 for parameters and performance.

6.4 Performance and Parameters

Tab. 3 gives an overview of the geometric complexity of the tetrahedral meshes used in the experiments. The complexity of the simulation scenarios varies from 700 tetrahedrons to 3700 tetrahedrons. Depending on the quality of the tetrahedrons and the additional computational costs for collision handling and visualization up to 5000 tetrahedrons can be handled at interactive rates. Note, that the numerical integration process itself is capable of handling up to 25000 tetrahedrons at interactive rates (see Tab. 1).

Tab. 4 gives an overview of the geometric complexity of the surface meshes that are coupled to the tetrahedral meshes for visualization purposes. Here we use meshes of up to 50000 faces which add a high-quality visual feedback to the simulation.

Tab. 5 shows all parameters for the experiments presented in this paper. The most relevant parameter for ma-

setup	Fig.	points	n_D	n_A	n_V
snakes	4	928	3548	0	1764
pitbull	5	314	1287	0	700
cows	6	1002	4626	0	2916
cloth	7	1301	7500	2500	3700
face	8	472	2622	874	1277
dragon	10	343	1472	0	834

Table 3: Complexity of the models used for experiments. The number of mass points is given for each tetrahedral model. Further, n_D, n_A, n_V denote the numbers of considered distance-preserving, area-preserving, and volume-preserving forces per integration step, respectively. In case of $n_A = 0$, the object surface is not considered in the deformation model (see Sec. 3.3).

setup	Fig.	surface points	surface faces
snakes	4	46456	88872
pitbull	5	12520	25030
cows	6	8709	17376
cloth	7	1301	2500
face	8	472	874
dragon	10	5202	10001

Table 4: Complexity of the surface meshes visualized in the experiments. The number of vertices and faces is given.

terial properties is k_D. Larger values as used in the "pitbull" experiment result in a stiff material, while smaller values as used in the "dragon" experiment result in soft or even fluid-like material.

The stiffness coefficient k_V is responsible for the volume preservation. This parameter also avoids inverted tetrahedrons. In most experiments, this parameter is set to avoid the inversion of tetrahedrons. In the "dragon" experiment, k_V is significantly larger than k_D which results in volume preservation, while distances between mass points are not preserved.

The parameter k_A is only used for discrete shells and thin plates. In the case of deformable triangle meshes it is difficult to map values of individual parameters to certain properties such as resistance against stretch, shearing, or bending. We have not further investigated the correlation between the stiffness coefficients and material properties for triangulated meshes.

The damping coefficient k_d improves the stability of dynamic simulations. As mentioned in Sec. 3.1, damping is only applied to distance-preserving forces, thus reducing internal oscillations of mass points. Experiments have shown, that there exist optimal values for k_d. If k_d is too large, energy is unintentionally added to the simulation.

Tab. 6 gives an performance overview for all experiments. This table illustrates, that the time step for the integration is usually similar to the computational time required to compute this time step. If the ratio of both

setup	Fig.	k_D	k_A	k_V	k_d
snakes	4	30	0	40	0.0001
pitbull	5	100	0	1	0.0001
cows	6	20	0	10	0.001
cloth	7	10	10	10	0.001
face	8	5	1	1	0.001
dragon	10	0.01	0	1	0.0001
cubes	3	10	0	20	0.001

Table 5: Stiffness coefficients k_D, k_A, k_V in Nm and damping coefficient k_d.

values is one, the simulation runs at real-time. The given computational times are used for integration and collision handling. In cases with more than one tetrahedral mesh, collision handling is comparatively expensive and consumes up to 50 % of the computational time. In other environments with only one deformable object, collision handling is only performed for planar walls and a sphere, which is negligible. The performance of the visualization obviously depends on the complexity of our surface meshes. If the visualization is too expensive, we usually perform several simulation steps until the scene is rendered. However, a rendering rate of more than 20 Hz is always guaranteed.

Fig.	time step [ms]	comp. time [ms]	integr.	coll.	vis. [ms]
4	4.0	6.36	66%	34%	19.4
5	2.7	1.44	99%	1%	5.5
6	7.1	8.41	57%	43%	4.6
7	2.6	7.45	99%	1%	0.8
8	2.9	2.29	99%	1%	0.3
10	1.7	1.57	99%	1%	2.6

Table 6: Performance measurements (Intel Pentium 4, 2.8 GHz, 1GB RAM, GeForce 4 Ti 4600). Time steps for the numerical integration are given. Further, computational times for integration, collision handling, and visualization are given. The computational time is used for integration and collision handling with the given percentages.

7 Conclusion

We have presented a new model that can be used to represent deformable tetrahedral meshes and deformable triangle meshes. The model considers elastic and plastic deformation. It handles a large variety of material properties ranging from stiff to fluid-like behavior. The proposed model extends existing deformable modeling techniques by incorporating efficient ways for volume and surface area preservation.

The computational efficiency of our approach is similar to simple mass-spring systems. Thus, environments of up to several thousand deforming primitives can be handled at interactive speed. Experiments have been described to show the capabilities of our simulation system with integrated collision handling.

Ongoing work focusses on the integration of the presented deformable model into computational surgery. First projects investigate potential applications in hysteroscopy simulation and simulation of stent placement.

Acknowledgements

This research is supported by the Swiss National Science Foundation and by the Swiss National Commission for Technology and Innovation KTI. The project is part of the Swiss National Center of Competence in Research on Computer Aided and Image Guided Medical Interventions (NCCR Co-Me). Further, it is part of KTI project no. 6310.1 KTS-ET.

References

[1] D. Baraff, A. Witkin, “Dynamic Simulation of Non-penetrating Flexible Bodies,” *Computer Graphics*, vol. 26, no. 2, pp. 303-308, 1992.

[2] D. Baraff, A. Witkin, “Large Steps in Cloth Simulation,” *Proc. of SIGGRAPH’98,* Orlando, Florida, pp. 43-54, 1998.

[3] D. Beeman, “Some Multistep Methods for use in Molecular Dynamics Calculations,” *Journal of Computational Physics,* vol. 20, pp. 130-139, 1976.

[4] R. Bridson, R. Fedkiw, J. Anderson, “Robust treatment of collisions, contact and friction for cloth animation,” *Proc. of SIGGRAPH’02,* San Antonio, Texas, pp. 594-603, 2002.

[5] E. Caramana, D. Burton, M. Shashkov, P. Whalen, “The Construction of Compatible Hydrodynamics algorithms Utilizing Conservation of Total Energy,” *Journal of Computational Physics,* vol. 146, pp. 227-262, 1998.

[6] J. Chadwick, D. Haumann, R. Parent, “Layered Construction for Deformable Animated Characters,” *Proc. of SIGGRAPH’89,* Boston, Massachusetts, pp. 243-252, 1989.

[7] G. Debunne, M. Desbrun, M.-P. Cani, A. Barr, “Adaptive Simulation of Soft Bodies in Real-Time,” *Proc. of Symposium on Computer Animation,* Philadelphia, Pennsylvania, pp. 133-144, 2000.

[8] G. Debunne, M. Desbrun, M.-P. Cani, A. Barr, “Dynamic Real-Time Deformations Using Space and Time Adaptive Sampling,” *Proc. SIGGRAPH’01,* Los Angeles, California, pp. 31-36, 2001.

[9] B. Eberhardt, A. Weber, W. Strasser, “A Fast, Flexible Particle-System Model for Cloth Draping,” *IEEE Computer Graphics and Applications,”* vol. 16, no. 5, pp. 52-59, 1996.

[10] A. Fuhrmann, C. Gross, V. Luckas, “Interactive Animation of Cloth Including Self Collision Detection,” *Proc. of WSCG’03,* University of West Bohemia, Czech Republic, pp. 141-148, 2003.

[11] C. Gear, *“Numerical Initial Value Problems in Ordinary Differential Equations,”* Prentice Hall Inc., Englewood Cliffs, New Jersey, 1971.

[12] S. Gibson, B. Mitrich, *“A Survey of Deformable Models in Computer Graphics,”* Technical Report TR-97-19, Mitsubishi Electric Research Laboratories MERL, Cambridge, Massachusetts, 1997.

[13] E. Grispun, P. Krysl, P. Schröder, “CHARMS: A Simple Framework for Adaptive Simulation,” *Proc. of SIGGRAPH’02,* San Antonio, Texas, pp. 281-290, 2002.

[14] E. Grispun, A. Hirani, M. Desbrun, P. Schröder, “Discrete Shells,” *Proc. of Symposium on Computer Animation,* San Diego, California, pp. 62-67, 2003.

[15] M. Hauth, O. Etzmuss, B. Eberhardt, R. Klein, R. Sarlette, M. Sattler, K. Daubert, J. Kautz, “Cloth Animation and Rendering,” *Eurographics Tutorials,* 2002.

[16] R. Hockney, “The Potential Calculation and Some Applications,” B. Alder, S. Fernbach, M. Rotenberg (eds.): *Methods in Computational Physics,* Plasma Physics, Academic Press, New York, vol. 9, pp. 136-211, 1970.

[17] D. James, D. Pai, “Artdefo. Accurate Real-Time Deformable Objects,” *Proc. of SIGGRAPH’99,* Los Angeles, California, pp. 65-72, 1999.

[18] Z. Kacic-Alesic, M. Nordenstam, D. Bullock, “A Practical Dynamics System,” *Proc. of Symposium on Computer Animation,* San Diego, California, pp. 7-16, 2003.

[19] T. Milliron, R. Jensen, R. Barzel, A. Finkelstein, “A Framework for Geometric Warps and Deformations,” *ACM Transactions on Graphics*, vol. 21, no. 1, pp. 20-51, 2002.

[20] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, B. Cutler, “Stable Real–Time Deformations,” *Proc. of Symposium on Computer Animation,* San Antonio, Texas, pp. 49-54, 2002.

[21] J. O’Brien, A. Bargteil, J. Hodgins, “Graphical Modeling and Animation of Ductile Fracture,” *Proc. of SIGGRAPH’02,* San Antonio, Texas, pp. 291-294, 2002.

[22] X. Provot, “Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior,” *Graphics Interface,* pp. 147-154, 1995.

[23] W. Swope, H. Andersen, P. Berenc, K. Wilson, “A Computer Simulation Method for the Calculation of Equilibrium Constants for the Formation of Physical Clusters of Molecules: Application to Small Water Clusters,” *Journal of Chemical Physics,”* vol. 76, no. 1, 1982.

[24] D. Terzopoulos, J. Platt, A. Barr, K. Fleischer, “Elastically Deformable Models,” *Proc. of SIGGRAPH’87,* Anaheim, California, pp. 205-214, 1987.

[25] D. Terzopoulos, K. Fleischer, “Deformable Models,” *The Visual Computer,* vol. 4, pp. 306-331, 1988.

[26] D. Terzopoulos, K. Fleischer, “Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture,” *Proc. of SIGGRAPH’88,* Atlanta, Georgia, pp. 269-278, 1988.

[27] M. Teschner, B. Heidelberger, D. Pomeranets, M. Müller, M. Gross, “Optimized Spatial Hashing for Collision Detection of Deformable Objects,” *Proc. of Vision, Modeling, Visualization,* Munich, Germany, pp. 47-54, 2003.

[28] L. Verlet, “Computer Experiments on Classical Fluids. Ii. Equilibrium Correlation Functions,” *Physical Review,* vol. 165, pp. 201-204, 1967.

[29] P. Volino, N. Magnenat-Thalmann, “Comparing Efficiency of Integration Methods for Cloth Animation,” *Proc. of Computer Graphics International,* Hong Kong, pp. 265-274, 2001.

[30] X. Wu, M. Downes, T. Goktekin, F. Tendick, “Adaptive Nonlinear Finite Elements for Deformable Body Simulation Using Dynamic Progressive Meshes,” *Proc. of Eurographics’01,* Manchester, United Kingdom, pp. 349-358, 2001.