Volume 25 (2006), number 4 pp. 809–836

COMPUTER GRAPHICS *forum*

Physically Based Deformable Models in Computer Graphics

Andrew Nealen[1], Matthias Müller[2,3], Richard Keiser[3], Eddy Boxerman[4] and Mark Carlson[5]

[1] Computer Graphics, TU Berlin
andy@nealen.net
[2] NovodeX/AGEIA
mmüller@ageia.com
[3] Computer Graphics Lab, ETH Zürich
keiser@inf.ethz.ch
[4] Ubisoft Montreal
eddybox@gmail.com
[5] DNA Productions, Inc.
mark.t.carlson@gmail.com

Abstract

Physically based deformable models have been widely embraced by the Computer Graphics community. Many problems outlined in a previous survey by Gibson and Mirtich have been addressed, thereby making these models interesting and useful for both offline and real-time applications, such as motion pictures and video games. In this paper, we present the most significant contributions of the past decade, which produce such impressive and perceivably realistic animations and simulations: finite element/difference/volume methods, mass-spring systems, mesh-free methods, coupled particle systems and reduced deformable models-based on modal analysis. For completeness, we also make a connection to the simulation of other continua, such as fluids, gases and melting objects. Since time integration is inherent to all simulated phenomena, the general notion of time discretization is treated separately, while specifics are left to the respective models. Finally, we discuss areas of application, such as elasto-plastic deformation and fracture, cloth and hair animation, virtual surgery simulation, interactive entertainment and fluid/smoke animation, and also suggest areas for future research.

Keywords: physically based animation, deformation, continuum elasticity, time integration, FEM, mass-spring, mesh-free methods, modal analysis, fluid animation

ACM CCS: I.3.5 Computer Graphics: Physically based modeling I.3.7 Computer Graphics: Animation and virtual reality

1. Introduction

Physicallybased deformable models have two decades of history in Computer Graphics: since Lasseter's discussion of *squash and stretch* [1] and, concurrently, Terzopoulos *et. al's* seminal paper on elastically deformable models [2], numerous researchers have partaken in the quest for the visually and physically plausible animation of deformable objects and fluids. This inherently interdisciplinary field elegantly combines Newtonian dynamics, continuum mechanics, numerical computation, differential geometry, vector calculus, approximation theory and computer graphics (to name a few) into a vast and powerful tool kit, which is being further explored and extended. The field is in constant flux and, thus, active and fruitful, with many visually stunning achievements to account for.

Since Gibson and Mirtich's survey paper [3], the field of physically based deformable models in computer graphics has expanded tremendously. Significant contributions were made in many key areas, e.g., object modeling, fracture, plasticity, cloth animation, stable fluid simulation, time integration strategies, discretization and numerical solution of PDEs, modal analysis, space-time adaptivity, multiresolution

Submitted September 2005
Revised May 2006
Accepted May 2006

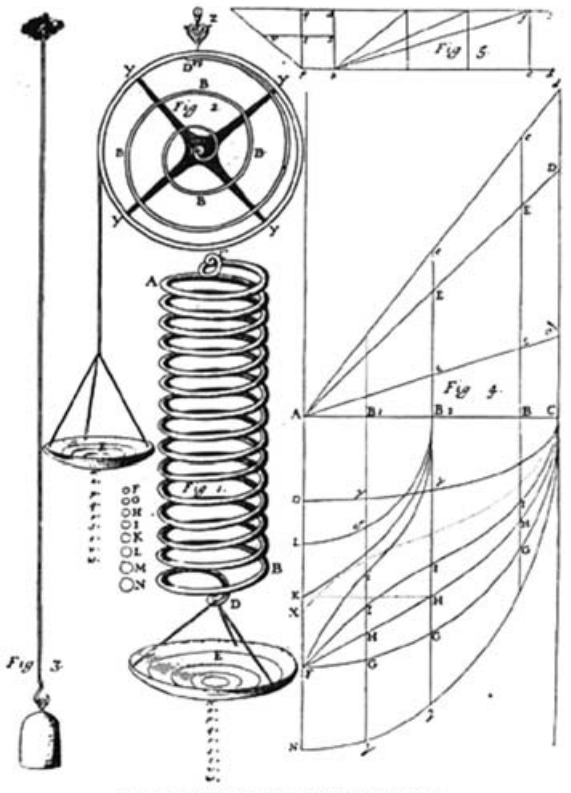

PLATE TO HOOKE'S LECTURE 'OF SPRING' 1678.

FIG. 1. Wire helical spring stretched to points r, p, q, r, s, t, v, w, by weights F, G, H, I, K, L, M, N.
FIG. 2. Watch spring similarly stretched by weights put in pan.
FIG. 3. The 'Springing of a string of Brass Wire 36 ft. long'.
FIG. 4. Diagram of velocities of springs.
FIG. 5. Diagram of law of ascent and descent of heavy bodies.

Figure 1: *Hooke's Law, from* De Potentia Restitutiva [1678].

modeling and real-time simulation. Nonphysical models, such as parametric curves and surfaces and free-form deformations, are not discussed in this paper. The inclined reader is therefore encouraged to browse more recent literature on t-splines [4, 5], space-warping [6–8] and methods based on differential surface properties [9–14]. For advances in character skinning see for example [15–17]. Since we are not able to cover basic elasticity theory and continuum mechanics in this paper, we would like to point out that a nice review of the history of elasticity theory, starting with the discovery of Hooke's Law in 1660 (Figure 1) and leading up to the general equations of Navier in 1821, is given in [18]. Furthermore, great introductions to continuum mechanics and dynamics can be found in [19] and in general textbooks, such as [20–24]. For application-specific presentations, we refer the reader to a number of recent works. For cloth simulation, there is the text by House and Breen [25], as well as the recent, extensive tutorial by Thalmann *et al.* [26]. For hair simulation, there is the (already slightly dated) overview by Thalmann *et al.* [27]; the paper by Volino and Thalmann [28] gives a good, more recent overview. Collision detection and haptic force-feedback rendering for deformable objects are other challenging and active areas of research. For a summary of recent work in these fields, we refer the reader to the report by Teschner *et al.* [29] and the course notes of Lin and Otaduy [30].

In this paper we take a *model-based* point of view, motivated by the fact that there are many readily available physical models for very similar applications, that is, we can animate an elastically or plastically deforming solid with many different underlying models, such as mass-spring systems, finite elements or mesh-free methods. We furthermore make a distinction between *Lagrangian* methods, where the model consists of a set of points with varying locations and properties, and *Eulerian* methods, where model properties are computed for a set of stationary points. To give a coarse overview, we describe recent developments for

- Lagrangian mesh-based methods
  - continuum mechanics based methods
  - mass-spring systems
- Lagrangian mesh free methods
  - loosely coupled particle systems
  - smoothed particle hydrodynamics (SPH)
  - mesh free methods for the solution of PDEs
- Reduced deformation models and modal analysis
- Eulerian and semi-Lagrangian methods
  - fluids and gases
  - melting objects

In each section we present the basic model formulation, recent contributions, benefits and drawbacks and various areas of application. The section on fluids, gases and melting objects contains an overview of recent work and establishes the connection to the field of physically based deformable models. A complete survey on the animation of fluids and gases would easily fill its own report and is therefore beyond our scope.

Our goal is to provide an up-to-date report to the Computer Graphics community, as an entry point for researchers and developers who are new to the field, thereby complementing the existing survey paper [3].

2. Background

2.1. Continuum elasticity

A deformable object is typically defined by its undeformed shape (also called equilibrium configuration, rest or initial shape) and by a set of material parameters that define how it

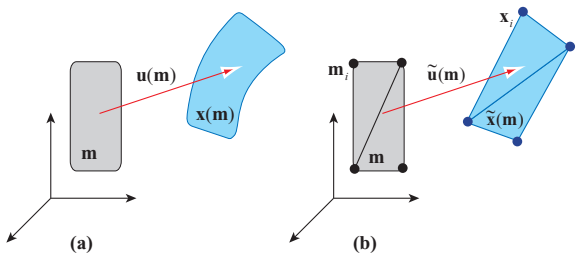

Figure 2: *In the finite element method, a continuous deformation (a) is approximated by a sum of (linear) basis functions defined inside a set of finite elements (b).*

deforms under applied forces. If we think of the rest shape as a continuous connected subset M of $\mathbb{R}^3$, then the coordinates $\mathbf{m} \in M$ of a point in the object are called *material coordinates* of that point. In the discrete case, M is a discrete set of points that sample the rest shape of the object.

When forces are applied, the object deforms and a point originally at location $\mathbf{m}$ (i.e., with material coordinates $\mathbf{m}$) moves to a new location $\mathbf{x}(\mathbf{m})$, the *spatial* or *world coordinates* of that point. Since new locations are defined for all material coordinates $\mathbf{m}$, $\mathbf{x}$ is a vector field defined on M. Alternatively, the deformation can also be specified by the *displacement* vector field $\mathbf{u}(\mathbf{m}) = \mathbf{x}(\mathbf{m}) - \mathbf{m}$ defined on M (see Figure 2). From $\mathbf{u}(\mathbf{m})$ the elastic strain ε is computed (ε is a dimensionless quantity that, in the (linear) 1D case, is simply $\Delta l/l$). A spatially constant displacement field represents a translation of the object with no strain. Therefore, it becomes clear that strain must be measured in terms of spatial variations of the displacement field $\mathbf{u} = \mathbf{u}(\mathbf{m}) = (u, v, w)^T$. Popular choices in computer graphics are

$$\varepsilon_G = \frac{1}{2}(\nabla\mathbf{u} + [\nabla\mathbf{u}]^T + [\nabla\mathbf{u}]^T\nabla\mathbf{u}) \tag{1}$$

$$\varepsilon_C = \frac{1}{2}(\nabla\mathbf{u} + [\nabla\mathbf{u}]^T) \tag{2}$$

where the symmetric tensor $\varepsilon_G \in \mathbb{R}^{3\times3}$ is Green's nonlinear strain tensor and $\varepsilon_C \in \mathbb{R}^{3\times3}$ its linearization, Cauchy's linear strain tensor. The gradient of the displacement field is a 3×3 matrix

$$\nabla\mathbf{u} = \begin{bmatrix} u_{,x} & u_{,y} & u_{,z} \\ v_{,x} & v_{,y} & v_{,z} \\ w_{,x} & w_{,y} & w_{,z} \end{bmatrix} \tag{3}$$

where the index after the comma represents a spatial derivative.

The material law (or *constitutive law*) is used for the computation of the symmetric internal stress tensor $\sigma \in \mathbb{R}^{3\times3}$ for each material point $\mathbf{m}$ based on the strain ε at that point (σ is measured as force per unit area, where 1 Pascal = 1 Pa = 1 N/m^2). Most computer graphics papers use Hooke's linear material law

$$\sigma = \mathbf{E} \cdot \varepsilon \tag{4}$$

where $\mathbf{E}$ is a rank four tensor that relates the coefficients of the stress tensor linearly to the coefficients of the strain tensor. For isotropic materials, the coefficients of $\mathbf{E}$ only depend on Young's modulus and Poisson's ratio.

2.2. Time integration

In order to simulate dynamic deformable solids, we need to know the time-dependent world coordinates $\mathbf{x}(\mathbf{m}, t)$ of all points in M. Given $\mathbf{x}(\mathbf{m}, t)$, we can subsequently display the configurations $\mathbf{x}(0)$, $\mathbf{x}(\Delta t)$, $\mathbf{x}(2\Delta t)$, .. resulting in an animation of the object. Here Δt is a fixed time step of the simulation and $\mathbf{x}(t)$ represents the entire vector field at time t.

The unknown vector fields $\mathbf{x}(t)$ are not given directly but implicitly as the solution of a differential equation, namely Newton's second law of motion of the form

$$\ddot{\mathbf{x}} = F(\dot{\mathbf{x}}, \mathbf{x}, t) \tag{5}$$

where $\ddot{\mathbf{x}}$ and $\dot{\mathbf{x}}$ are the second and first time derivatives of $\mathbf{x}$, respectively and $F(\cdot)$ is a general function given by the physical model of the deformable object. In order to find the solution $\mathbf{x}(t)$, this second-order differential equation is often rewritten as a coupled set of two first-order equations

$$\dot{\mathbf{x}} = \mathbf{v} \tag{6}$$

$$\dot{\mathbf{v}} = F(\mathbf{v}, \mathbf{x}, t) \tag{7}$$

where the new quantity $\mathbf{v}$ represents $\dot{\mathbf{x}}$. A discrete set of values $\mathbf{x}(0)$, $\mathbf{x}(\Delta t)$, $\mathbf{x}(2\Delta t)$, .. of the unknown vector field $\mathbf{x}$ which is needed for the animation can now be obtained by solving (i.e. integrating) this system of equations numerically. Numerical integration of ordinary differential equations (ODEs) is the subject of many textbooks (e.g. [31,32]). See [33] for an excellent overview in the context of deformable modeling in computer graphics. We give a few examples here, which appear in subsequent sections.

The simplest scheme is explicit (or forward) Euler integration, where the time derivatives are replaced by finite differences $\dot{\mathbf{v}}(t) = [\mathbf{v}(t + \Delta t) - \mathbf{v}(t)]/\Delta t$ and $\dot{\mathbf{x}}(t) = [\mathbf{x}(t + \Delta t) - \mathbf{x}(t)]/\Delta t$. Substituting these into the above equations and solving for the quantities at the next time step $t + \Delta t$ yields

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t\mathbf{v}(t) \tag{8}$$

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t F(\mathbf{v}(t), \mathbf{x}(t), t) \tag{9}$$

Time integration schemes are evaluated by two main criteria, their stability and their accuracy. Their accuracy is measured by their convergence with respect to the time step size Δt, that is, first-order $O(\Delta t)$, second-order $O(\Delta t^2)$, etc. In the field of physically based animation in computer graphics, stability is often much more important than accuracy.

The integration scheme presented above is called *explicit* because it provides explicit formulas for the quantities at the next time step. Explicit methods are easy to implement but they are only conditionally stable, that is, stable only if Δt is smaller than a stability threshold (see [34] for an example). For stiff objects this threshold can be very small. The instability is due to the fact that explicit methods extrapolate a constant right-hand side blindly into the future as the above equations show. For a simple spring and a large Δt, the scheme can overshoot the equilibrium position arbitrarily. At the next time step the restoring forces get even larger resulting in an exponential gain of energy and finally an explosion. This problem can be solved by using an *implicit* scheme that uses quantities at the next time step $t + \Delta t$ on both sides of the equation

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \mathbf{v}(t + \Delta t) \tag{10}$$

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t F(\mathbf{v}(t + \Delta t), \mathbf{x}(t + \Delta t), t) \tag{11}$$

The scheme is now called implicit because the unknown quantities are *implicitly* given as the solution of a system of equations. Now, instead of extrapolating a constant right-hand side blindly into the future, the right-hand side is part of the solution process. Remarkably, the implicit (or backward) Euler scheme is stable for arbitrarily large time steps Δt. (There is, however, a lower time step limit that, for practical purposes, poses no problem). This gain comes with the price of having to solve an algebraic system of equations at each time step (linear if $F(\cdot)$ is linear, nonlinear otherwise).

A simple improvement to the forward Euler scheme is to swap the order of the equations and use a forward–backward scheme

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t F(\mathbf{v}(t), \mathbf{x}(t), t) \tag{12}$$

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \mathbf{v}(t + \Delta t) \tag{13}$$

The update to $\mathbf{v}$ uses forward Euler, while the update to $\mathbf{x}$ uses backward Euler. Note that the method is still explicit; $\mathbf{v}(t + \Delta t)$ is simply evaluated first. For nondissipative systems (i.e., when forces are independent of velocities), this reduces to the second-order accurate Stoermer–Verlet scheme. The forward–backward Euler scheme is more stable than standard forward Euler integration, without any additional computational overhead.

3. Lagrangian Mesh-Based Methods

3.1. The finite element method

The finite element method (FEM) is one of the most popular methods in computational sciences to solve partial differential equations (PDEs) on irregular grids. In order to use the method for the simulation of deformable objects, the object is viewed as a continuous connected volume as in Section 2.1, which is discretized using an irregular mesh. Continuum mechanics, then, provides the PDE to be solved.

The PDE governing dynamic elastic materials is given by

$$\rho \ddot{\mathbf{x}} = \nabla \cdot \sigma + \mathbf{f} \tag{14}$$

where ρ is the density of the material and $\mathbf{f}$ externally applied forces such as gravity or collision forces. The divergence operator turns the 3×3 stress tensor back into a 3 vector

$$\nabla \cdot \sigma = \begin{bmatrix} \sigma_{xx,x} + \sigma_{xy,y} + \sigma_{xz,z} \\ \sigma_{yx,x} + \sigma_{yy,y} + \sigma_{yz,z} \\ \sigma_{zx,x} + \sigma_{zy,y} + \sigma_{zz,z} \end{bmatrix} \tag{15}$$

representing the internal force resulting from a deformed infinitesimal volume. Equation 14 shows the equation of motion in *differential form* in contrast to the *integral form*, which is used in the finite volume method.

The FEM is used to turn a PDE into a set of algebraic equations that are then solved numerically. To this end, the domain M is discretized into a finite number of disjoint elements (i.e., a mesh). Instead of solving for the spatially continuous function $\mathbf{x}(\mathbf{m}, t)$, one only solves for the discrete set of unknown positions $\mathbf{x}_i(t)$ of the nodes of the mesh. First, the function $\mathbf{x}(\mathbf{m}, t)$ is approximated using the nodal values by

$$\tilde{\mathbf{x}}(\mathbf{m}, t) = \sum_i \mathbf{x}_i(t) \mathbf{b}_i(\mathbf{m}) \tag{16}$$

where $\mathbf{b}_i(\cdot)$ are fixed nodal basis functions that are 1 at node i and zero at all other nodes, also known as the Kronecker Delta property (see Figure 2). In the most general case of the FEM, the basis functions do not have this property. In that case, the unknowns are general parameters that cannot be interpreted as nodal values. Substituting $\tilde{\mathbf{x}}(\mathbf{m}, t)$ into Equation 14 results in algebraic equations for the $\mathbf{x}_i(t)$. In the Galerkin approach [35], finding the unknowns $\mathbf{x}_i(t)$ is viewed as an optimization process. When substituting $\mathbf{x}(\mathbf{m}, t)$ by the approximation $\tilde{\mathbf{x}}(\mathbf{m}, t)$, the infinitely dimensional search space of possible solutions is reduced to a finite dimensional subspace. In general, no function in that subspace can solve the original PDE. The approximation will generate a deviation or residue when substituted into the PDE. In the Galerkin method, the approximation that *minimizes the residue* is sought. In other words, we look for an approximation whose residue is perpendicular to the subspace of functions defined by Equation 16.

Many papers in computer graphics use a simple form of the FEM for the simulation of deformable objects, sometimes called the *explicit* FEM, which is quite easy to understand and to implement; instead of solving a system of equations for the positions $\mathbf{x}_i(t)$, nodal forces are computed locally from adjacent elements [36–38]. The explicit FEM is not to be confused with the standard FEM being *integrated* explicitly. The explicit FEM can be integrated either explicitly or implicitly. In the explicit finite element approach, both, the masses and the internal and external forces are lumped to the vertices. The nodes in the mesh are treated like mass

points in a mass-spring system while each element acts like a generalized spring connecting all adjacent mass points. The forces acting on the nodes of an element due to its deformation are computed as follows (see for instance [36]): given the positions of the vertices of an element and the fixed basis functions, the continuous deformation field $\mathbf{u}(\mathbf{m})$ inside the element can be computed using Equation 16. From $\mathbf{u}(\mathbf{m})$, the strain field $\varepsilon(\mathbf{m})$ and stress field $\sigma(\mathbf{m})$ are computed. The deformation energy of the element is then given by

$$E = \int_V \varepsilon(\mathbf{m}) \cdot \sigma(\mathbf{m}) d\mathbf{m} \tag{17}$$

where the dot represents the componentwise scalar product of the two tensors. The forces can then be computed as the derivatives of the energy with respect to the nodal positions. In general, the relationship between nodal forces and nodal positions is nonlinear. When linearized, the relationship for an element e connecting n_e nodes can simply be expressed as

$$\mathbf{f}_e = \mathbf{K}_e \, \mathbf{u}_e \tag{18}$$

where $\mathbf{f}_e \in \mathbb{R}^{3n_e}$ contains the n_e nodal forces and $\mathbf{u}_e \in \mathbb{R}^{3n_e}$ the n_e nodal displacements of an element. The matrix $\mathbf{K}_e \in \mathbb{R}^{3n_e \times 3n_e}$ is called the *stiffness matrix* of the element. Because elastic forces coming from adjacent elements add up at a node, a stiffness matrix $\mathbf{K} \in \mathbb{R}^{3n \times 3n}$ for an entire mesh with n nodes can be formed by assembling the element's stiffness matrices

$$\mathbf{K} = \sum_e \mathbf{K}_e \tag{19}$$

In this sum, the element's stiffness matrices are expanded to the dimension of $\mathbf{K}$ by filling in zeros at positions related to nodes not adjacent to the element. Using the linearized elastic forces, the linear algebraic equation of motion for an entire mesh becomes ($\mathbf{u} = \mathbf{x} - \mathbf{x}_0$)

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{D}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f}_{ext} \tag{20}$$

where $\mathbf{M} \in \mathbb{R}^{n \times n}$ is the mass matrix, $\mathbf{D} \in \mathbb{R}^{n \times n}$ the damping matrix and $\mathbf{f}_{ext} \in \mathbb{R}^n$ externally applied forces. Often, diagonal matrices are used for $\mathbf{M}$ and $\mathbf{D}$, a technique called *mass lumping*. In this case, $\mathbf{M}$ just contains the point masses of the nodes of the mesh on its diagonal. The vectors $\mathbf{x}$ and $\mathbf{x}_0$ contain, respectively, the actual and the rest positions of the nodes.

The finite element method only produces a *linear* system of algebraic equations if applied to a *linear* PDE. If a linear strain measure is used and Hooke's law for isotropic materials is substituted into 14, Lamé's linear PDE results

$$\rho\ddot{\mathbf{x}} = \mu\Delta\mathbf{u} + (\lambda + \mu)\nabla(\nabla \cdot \mathbf{u}) \tag{21}$$

where the material constants λ and μ can be computed directly from Young's modulus and Poisson's ratio. This equation is solved in [39] in a multiresolution fashion. Using discretized versions of the Laplacian ($\Delta = \nabla^2$) and gradient-of-divergence ($\nabla(\nabla \cdot)$) operators, they solve the Lamé equation on an irregular, multiresolution grid. The system is optimized for limited deformations (linearized strain) and does not support topological changes. Based on Gauss' divergence theorem, the discrete operators are further evolved in [40], which leads to greater accuracy through defined error bounds. Furthermore, the cubic octree hierarchy employed in [39] is succeeded by a non-nested hierarchy of meshes, in conformance with the redefined operators, which leads to improved shape sampling. In [37] the previous linearized physical model is replaced by local explicit finite elements and Green's nonlinear strain tensor. To increase stability the simulation is integrated semi-implicitly in time [41].

O'Brien *et al.* [36] and [42] present a FEM-based technique for simulating brittle and ductile fracture in connection with elastoplastic materials. They use tetrahedral meshes in connection with linear basis functions $\mathbf{b}_i(\cdot)$ and Green's strain tensor. The resulting nonlinear equations are solved explicitly and integrated explicitly. The method produces realistic and visually convincing results, but it is not designed for interactive or real-time use. In addition to the strain tensor, they use the so-called *strain rate tensor* (the time derivative of the strain tensor), to compute damping forces. Other studies on the visual simulation of brittle fracture are [43] and [44]. Bro-Nielsen and Cotin [45] use linearized finite elements for surgery simulation. They achieve significant speedup by simulating only the visible surface nodes (condensation), similar to the Boundary element method (BEM) (see Section 3.4). Many other studies successfully apply the FEM to surgery simulation, such as (but surely not limited to) [46–54].

As long as the equation of motion is integrated explicitly in time, nonlinear elastic forces resulting from Green's strain tensor pose no computational problems. The nonlinear formulas for the forces are simply evaluated and used directly to integrate velocities and positions as in [36]. As mentioned earlier (Section 2.2), explicit integration schemes are stable only for small time steps while implicit integration schemes allow arbitrarily large time steps. However, in the latter case, a system of algebraic equations needs to be solved at every time step. Linear PDEs yield linear algebraic systems that can be solved more efficiently and more stably than nonlinear systems. Unfortunately, linearized elastic forces are only valid for small deformations. Large rotational deformations yield highly inaccurate restoring forces (see Figure 3).

To eliminate these artifacts, Müller *et al.* extract the rotational part of the deformation for each finite element and compute the forces with respect to the nonrotated reference frame [55]. The linear Equation (18) for the elastic forces of an element (in this case a tetrahedron) is replaced by

$$\mathbf{f} = \mathbf{R}\mathbf{K}\left(\mathbf{R}^T\mathbf{x} - \mathbf{x}_0\right) \tag{22}$$

where $\mathbf{R} \in \mathbb{R}^{12 \times 12}$ is a matrix that contains four 3×3 identical rotation matrices along its diagonal. The vector $\mathbf{x}$ contains the actual positions of the four adjacent nodes of the tetrahedron while $\mathbf{x}_0$ contains their rest positions. The rotation of the element used in $\mathbf{R}$ is computed by performing a

Figure 3: *The pitbull with its inflated head (left) shows the artifact of linear FEM under large rotational deformations. The correct deformation is shown on the right.*

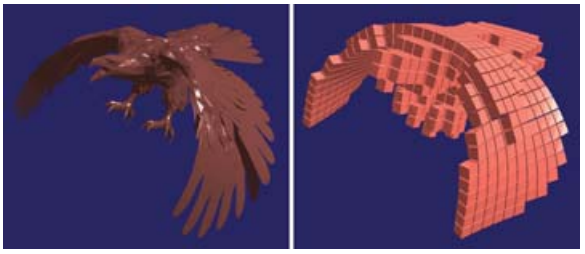

Figure 4: *Embedding a topologically inconsistent surface mesh in hexahedral finite elements for deformation simulation and fracturing [61].*

polar decomposition of the matrix that describes the transformation of the tetrahedron from the configuration $\mathbf{x}_0$ to the configuration $\mathbf{x}$. This yields stable, fast and visually pleasing results. In an earlier approach, they extract the rotational part not per element but per node [38]. In this case, the global stiffness matrix does not need to be reassembled at each time step but ghost forces are introduced.

Another solution to this problem is proposed in [56]: each region of the finite element mesh is associated with the bone of a simple skeleton and then locally linearized. The regions are blended in each time step, leading to results that are visually indistinguishable from the nonlinear solution, yet much faster.

An adaptive nonlinear FEM simulation is described by Wu *et al.* [57]. Distance, surface and volume preservation for triangular and tetrahedral meshes is outlined in [58]. Grinspun *et al.* introduce conforming, hierarchical, adaptive refinement methods (CHARMS) for general finite elements [59], refining basis functions instead of elements. Irving *et al.* [60] present a method that robustly handles large deformation and element inversion by computing a problem-specific diagonalization of the deformation gradient, from which the forces are derived. Müller *et al.* [61] propose simulating and fracturing objects represented by surface meshes, by embedding the surface in a cuboid element voxelization (Figure 4). This domain-embedding strategy is also used by James *et al.* in their *squashing cubes* simulator [62]. To support topological changes while maintaining well-shaped elements, Molino *et al.* create duplicates of the original elements in their virtual node algorithm [63].

3.2. The method of finite differences

If the object M is sampled using a *regular* spatial grid, the equation of motion 14 can be discretized using finite differences. The method of finite differences is easier to implement than the general FEM. However, it is difficult to approximate the boundary of an arbitrary object with a regular mesh. Also, local adaptations are only possible with irregular meshes.

Pioneering work in the field of physically based animation was carried out by Terzopoulos and his co-workers. In their seminal paper [2] the dynamics of deformable models are computed from the potential energy stored in the elastically deformed body. For volumetric objects, they define the deformation energy as the weighted matrix norm of the difference between the metric tensors of the deformed and original shape, integrated over the entire continuum (two-and one-dimensional objects involve further weighted norms of second- and third-order tensors). The continuous variational (or directional) derivative of this energy functional (the elastic force) is discretized using the method of finite differences (FD), and the simulation is moved forward through time by semi-implicit integration. This work is further evolved in [64] and [65] where the model is extended to cover plasticity and fracture.

3.3. The finite volume method

In the explicit FEM, the forces acting on the nodes of an element are computed as the derivatives of the deformation energy with respect to the nodal positions, where the deformation energy is computed via 17.

There is a more direct way to get to the nodal forces of an element, using the method of finite volumes. The stress tensor σ can be used to compute the internal force $\mathbf{f}$ per unit area with respect to a certain plane orientation as

$$\mathbf{f} = \sigma \mathbf{n} \tag{23}$$

where the normalized vector $\mathbf{n}$ is the normal on that plane. Thus, the total force acting on the face A of a finite element is given by the surface integral

$$\mathbf{f}_A = \int_A \sigma d\mathbf{A} \tag{24}$$

When linear basis functions are used, the stress tensor is constant within an element. Then, for planar element faces, the surface integral reduces to the simple product

$$\mathbf{f}_A = A\sigma \mathbf{n} \tag{25}$$

where the scalar A is the area of the face and $\mathbf{n}$ its normal. To get the nodal forces, one loops over all the faces A_i of

the element and distributes the force $\mathbf{f}_{A_i}$ evenly among the nodes adjacent to that face. Teran *et al.* [66] use this method to simulate skeletal muscle. They also use a geometrically motivated way to compute strain, which leads to an intuitive way of integrating the equations of motion.

3.4. The boundary element method

The boundary element method (BEM) is an interesting alternative to the standard finite element approach because all computations are done on the surface (boundary) of the elastic body instead of on its volume (interior). For a very good introduction to the subject see [35]. Roughly speaking, the integral form of the equation of motion is transformed into a surface integral by applying the Green–Gauss theorem. The method achieves substantial speedup because the three-dimensional problem is reduced to two dimensions. However, the approach only works for objects whose interior is composed of a homogenous material. Also, topological changes (e.g., fractures) are more difficult to handle than in the explicit FEM approach where only local changes to the stiffness matrix or the connectivity of the elements need to be done.

In the ArtDefo system [67] volumetric models are simulated using the BEM and a database of precomputed reference boundary value problems (RBVPs). By employing a fast update process, which exploits coherence, accurate real-time deformation simulation is achieved. In [68] the RBVPs are expressed in terms of linear elastostatic Green's functions (LEGFMs) and multiple RBVP's are linked via interface boundary conditions, resulting in a multizone elastokinematic model, which properly simulates large nonlinear relative strains. The system is further augmented by multiresolution Green's functions (wavelet Green's functions) in [69], with which large-scale objects can be simulated.

3.5. Mass-spring systems

Mass-spring systems are arguably the simplest and most intuitive of all deformable models. Instead of beginning with a PDE such as Equation (14) and subsequently discretizing in space, we begin directly with a discrete model. As the name implies, these models simply consist of point masses connected together by a network of massless springs.

The state of the system at a given time t is defined by the positions $\mathbf{x}_i$ and velocities $\mathbf{v}_i$ of the masses $i = 1, \ldots, n$. The force $\mathbf{f}_i$ on each mass is computed due to its spring connections with its neighbors, along with external forces such as gravity, friction, etc. The motion of each particle is then governed by Newton's second law $\mathbf{f}_i = m_i \ddot{\mathbf{x}}_i$, which for the entire particle system can be expressed as

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{v}) \tag{26}$$

where $\mathbf{M}$ is a $3n \times 3n$ diagonal mass matrix. Thus, mass-spring systems "only" require the solution of a system of coupled ODEs. This is done via a numerical integration scheme as described inSection 2.2.

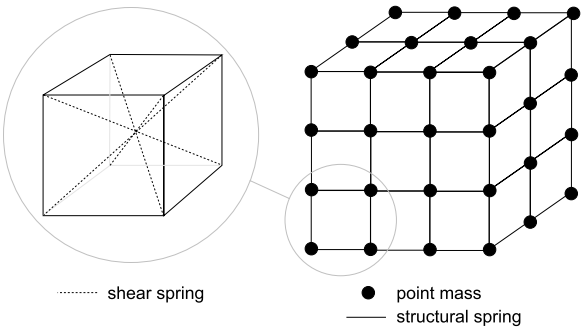

Figure 5: *A mass-spring system.*

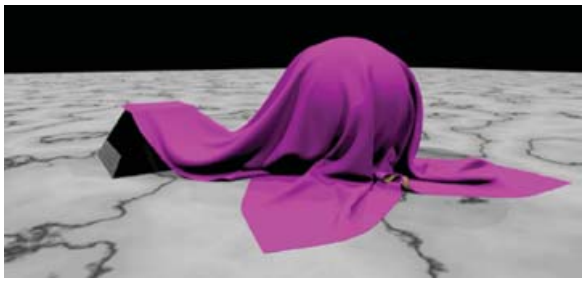

Figure 6: *Cloth carefully modeled using a mass-spring system. Image courtesy of Robert Bridson, UBC.*

Depicted in Figure 5 is a simple example of a volumetric mass-spring system (similar to that in [70]). The masses are regularly spaced in a lattice. They are connected together along the 12 edges of each hexahedron by "structural" springs. Masses on opposite corners of each hexahedron are connected together by "shear" springs. These springs cause the solid to resist longitudinal and shear deformations respectively, and their rest lengths define the undeformed state of the body. Typically, the spring types have different properties; and for anisotropic materials each springs' properties also depend on its orientation.

Springs are commonly modeled as being linearly elastic; the force acting on mass i, generated by a spring connecting i and j together is

$$\mathbf{f}_i = k_s(|\mathbf{x}_{ij}| - l_{ij})\frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|} \tag{27}$$

where $\mathbf{x}_{ij}$ is the difference between the two masses' position vectors $(\mathbf{x}_j - \mathbf{x}_i)$, k_s is the spring's stiffness and l_{ij} is its rest length.

Physical bodies are not perfectly elastic; they dissipate energy during deformation. To account for this, viscoelastic springs are used to damp out relative motion. Thus, in addition

to (27), each spring exerts a viscous force. It is common to model this as

$$\mathbf{f}_i = k_d(\mathbf{v}_j - \mathbf{v}_i) \tag{28}$$

where k_d is the spring's damping constant. This is unfortunate, as it damps rigid body rotations. Worse still, when modeling cloth it damps bending and wrinkling—a key feature of these objects. It is preferable to use

$$\mathbf{f}_i = k_d\left(\frac{\mathbf{v}_{ij}^T\mathbf{x}_{ij}}{\mathbf{x}_{ij}^T\mathbf{x}_{ij}}\right)\mathbf{x}_{ij} \tag{29}$$

where $\mathbf{v}_{ij} = \mathbf{v}_j - \mathbf{v}_i$. This *projects* the velocity difference onto the vector separating the masses, and only allows a force along that direction.

In the literature, the concept of a mass-spring system is often more general than the canonical example given above. These models are still represented by point masses and a fixed connectivity, but the notion of a spring is generalized. Often the name "particle system" is used instead, which is perhaps a more fitting name. (Though they should be distinguished from the models found in Section 4.1, which have no fixed connectivity.) In any case, the term "mass-spring system" is very suggestive, and will most likely persist.

In more general particle systems, such as those found in [71,72,58], deformation energies are defined. These are often derived from soft constraints that are to be maintained in the model; given some constraint of the form $\mathbf{C}(\mathbf{x}) = 0$, an associated energy is defined as $\frac{k_s}{2}\mathbf{C}^T(\mathbf{x})\mathbf{C}(\mathbf{x})$. These energies are minimized at the model's rest state, and are used to enforce the preservation of mesh distances, angles, areas, volumes, etc. Particle forces are then computed as the derivatives of the energies with respect to the particle positions

$$\mathbf{f}_i = -\frac{\partial E}{\partial \mathbf{x}_i} \tag{30}$$

Each energy term typically involves only a few particles. In the case of a distance constraint, we are back to our simple, linear spring model described above. However, for other constraints and energies, we must imagine more general spring types: angular, bending, areal, volumetric, etc.

3.5.1. *Early work*

Mass-spring networks first saw use in computer graphics for facial modeling [73,74]. These early works solve static problems of the form (18). Soon after, dynamic models were introduced to model skin, fat and muscle [75–77].

The locomotion of simple creatures such as snakes, worms and fish [78,79] was simulated using mass-spring systems. In these systems, spring rest-lengths vary over time to simulate muscle actuation.

In work by Terzopoulos *et al.* [80] the equations of thermal conductivity are used to simulate heat transfer in a volumetric mass-spring system. Each spring's stiffness is determined by its temperature, which is set to zero once the melting threshold is exceeded. A discrete fluid model (similar to those presented in Section 4.1) is applied to particles for which all connecting springs have melted.

Breen *et al.* [71] presented a particle system to model cloth. They posited that cloth is a mechanism of warp and weft fibers — not a continuum — and that mass-spring systems are thus *more* appropriate than finite element techniques for modeling cloth. Using measured data from the *Kawabata Evaluation System* [81], they took care in formulating their energy functions for stretching, bending and trellising (shearing), and predicted the static drape of real materials quite accurately. Since [71], particle systems have dominated the cloth simulation literature—although new FEM formulations (such as the one described in [82]) continue to be proposed.

3.5.2. *Drawbacks and improvements*

Particle systems tend to be intuitive and simple to implement. Moreover, they are computationally efficient and handle large deformations with ease. However, unlike the finite element and finite difference methods, which are built upon elasticity theory, mass-spring systems are not necessarily accurate. Primarily, most such systems are not convergent; that is, as the mesh is refined, the simulation does not converge on the true solution. Instead, the behavior of the model is dependent on the mesh resolution and topology. In practice, spring constants k_s are often chosen arbitrarily, and one can say little quantitatively about the material being modeled. In addition, there is often coupling between the various spring types. For instance, the "shear" springs of the model in Figure 5 also resist stretching. For medical applications, as well as for virtual garment simulation in the textile industry, greater accuracy is required. For applications such as film and games, this lack of accuracy may be acceptable; convincing animations have been produced using these systems. However, a judicious choice of model is still advised, as the behavior of the material being modeled can be highly mesh dependent.

Several researchers have explored and have tried to mitigate the accuracy issues in mass-spring systems. Kass [83] presents a simple equivalence in one dimension between a mass-spring system and a corresponding finite difference spatial discretization. Eischen and Bigliani [25] perform a comparison between a finite element model and a carefully tuned particle system, which gave similar experimental results for small deformations. Etzmuss *et al.* [84] carefully derive a cloth particle system from a continuous formulation by a finite difference discretization. Their model is convergent, and they show how to choose spring constants based on continuous material parameters.

Figure 7: *A bending model for triangle meshes with nonzero rest angles [87]. The model on the right has significantly stronger bending resistance, giving it a "water bottle" effect. Image courtesy of Robert Bridson, UBC.*

These more accurate particle systems, however, only apply to rectangular meshes. Like finite difference techniques, they do not generalize easily to triangular (or tetrahedral) meshes. Volino and Thalmann [85] present a triangular mesh that attempts to model the physical properties of cloth regardless of edge orientations. However, the accuracy of their method is unclear. Baraff and Witkin [72] model cloth using a triangular mesh, deriving in-plane particle forces from a continuum formulation. Their approach supports anisotropic behavior, but is not convergent. In [86], the authors present a novel mass-spring system that gives consistent results (though it is not convergent) for tetrahedral and hexahedral meshes, independent of the orientation of the elements. In their system, springs emanate from the barycenter of each element along principle coordinate axes and are attached to element faces; forces acting on each particle are then computed via interpolation. Their approach also allows the user to control anisotropic material behavior. Teschner *et al.* [58] model volumes and surfaces using tetrahedral and triangular meshes, respectively. They employ generalized springs that preserve distances, areas and volumes. Their model is efficient, convergent (though it does not incorporate continuous material properties) and supports plastic deformation.

Bending models for surfaces tend to be particularly *ad-hoc*; Bridson *et al.* [87] present a physically correct bending model for triangle meshes by isolating the bending mode from all other modes of deformation (Figure 7). Grinspun *et al.* present a similar bending model for the simulation of discrete shells [88].

An interesting approach is to use learning algorithms to *search* for mesh parameters. (See [89,90] and references therein.) Bhat *et al.* use simulated annealing to estimate the spring constants in a cloth mesh. They employ Baraff and Witkin's model [72], and use video of real cloth as their reference for comparison. Bianchi *et al.* use a genetic algorithm to identify spring constants as well as mesh topology in a volumetric mass-spring system. They compare their results to FEM reference solutions, and obtain close correspondence. While such methods are general, they are computationally expensive and require reference solutions; moreover, the results of a specific search may not hold for different mesh resolutions.

Adaptive meshing techniques are particularly complicated by the nonconvergent behavior of many mass-spring systems, as they *require* modeling at multiple resolutions. (Another difficulty is the handling of so-called T-junctions between regions of differing resolutions, though this problem must also be addressed in FEM and other methods.) A number of authors have addressed this issue for cloth simulation. In these systems, splitting/merging criterion is often based on bending angles between triangles. Villard and Borouchaki [91] apply adaptive refinement to a simple, convergent model based on quadrilateral meshes. They show consistent results between meshes of multiple resolutions, maintaining a rich detail at reduced computation times. Earlier work by Hutchinson *et al.* [92] takes a similar approach, though the convergence of their model is not clear as they couple their spatial and time discretizations together in an *ad-hoc* manner. Li and Volkov [93] present an adaptive framework for triangular cloth meshes, avoiding the problematic T-junctions. They present attractive results, though their physical model is not convergent.

Finally, a number of researchers have sought to improve the realism of particle systems by modeling nonlinear material properties. For instance, Eberhardt *et al.* [94] base their spring model on measured cloth data to model hysteresis effects; Choi and Ko [95] approximate cloth's buckling response using a fifth-order polynomial.

3.5.3. *Time integration*

Once a force model is chosen, the particle system is stepped forward in time via a numerical integration scheme as in Section 2.2. Time integration schemes have received particular attention in the cloth simulation literature (see [33]). As cloth is often modeled by mass-spring systems, we further discuss the subject of time integration here.

Cloth is a *highly* anisotropic material due to its structure: it resists bending weakly, but has a relatively strong resistance to stretching. When simulating a highly stiff volume of material, one can employ a reduced deformation model, such as modal analysis (see Section 5)—and in the limit, use a rigid body model. Clearly, this cannot be done with cloth. We are interested in visualizing the large-scale, out-of-plane folding and wrinkling of cloth, however we must still deal with these planar energies in our model. Numerically speaking, the resulting ODE system (26) is *stiff*; that is, it possesses a wide range of eigenvalues. This forces us to take excessively small time steps when using an explicit scheme.

Provot [96] tackles the issue of stiffness by using much weaker stretching energies and then post-processing the cloth mesh at each time step, iteratively enforcing constraints. Springs that are stretched by more than 10% are relaxed (shortened). This in turn stretches nearby neighboring springs that are then relaxed, and so on until convergence is obtained. This, in effect, also models the biphasic nature of cloth—small deformations are resisted weakly until a threshold

is reached, whereupon stiffness dramatically increases. In practice, this method gives efficient, attractive results, though convergence is not guaranteed.

Baraff and Witkin [72] present a linear implicit scheme that allows for large time steps while maintaining stability. This has proven to be a robust and efficient solution to the stiffness problem, and has become a popular technique. Specifically, they apply a backward Euler scheme to Equation (26), giving

$$\begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{v} \end{bmatrix} = \Delta t \begin{bmatrix} \mathbf{v}_n + \Delta\mathbf{v} \\ M^{-1}\mathbf{f}(\mathbf{x}_n + \Delta\mathbf{x}, \mathbf{v}_n + \Delta\mathbf{v}) \end{bmatrix} \quad (31)$$

where $\mathbf{x}_n = \mathbf{x}(t)$ and $\mathbf{v}_n = \mathbf{v}(t)$. This is a nonlinear equation in $\Delta\mathbf{x}$ and $\Delta\mathbf{v}$. A linear implicit version of (31) is obtained by using a first-order Taylor series expansion of $\mathbf{f}$

$$\mathbf{f}(\mathbf{x}_n + \Delta\mathbf{x}, \mathbf{v}_n + \Delta\mathbf{v}) = \mathbf{f}_n + \frac{\partial\mathbf{f}}{\partial\mathbf{x}}\Delta\mathbf{x} + \frac{\partial\mathbf{f}}{\partial\mathbf{v}}\Delta\mathbf{v} \quad (32)$$

where $\frac{\partial\mathbf{f}}{\partial\mathbf{x}}$ and $\frac{\partial\mathbf{f}}{\partial\mathbf{v}}$ are the Jacobian matrices of the particle forces with respect to position and velocity, respectively. In their paper, Baraff and Witkin derive expressions for the Jacobians of the various internal forces in their model. Due to the local connectivity structure of the mesh, these are sparse matrices. They then solve the resulting linear system at each time step

$$A\Delta\mathbf{v} = \mathbf{b} \quad (33)$$

where

$$A \equiv \left(I - \Delta t M^{-1}\frac{\partial\mathbf{f}}{\partial\mathbf{v}} - \Delta t^2 M^{-1}\frac{\partial\mathbf{f}}{\partial\mathbf{x}} \right) \quad (34)$$

$$\text{and} \quad \mathbf{b} \equiv \Delta t M^{-1}\left(\mathbf{f}_n + \Delta t \frac{\partial\mathbf{f}}{\partial\mathbf{x}}\mathbf{v}_n \right) \quad (35)$$

They solve this system, in the presence of constraints, using a modified preconditioned conjugate gradient solver (MPCG, an interesting technique in its own right—see [72,97]). This is equivalent to applying one Newton iteration to Equation 31. Although solving this system increases the computational cost at each time step, this is more than offset by the ability to take large steps when desired. However, as illustrated by Desbrun *et al.* [41], the larger the time step size, the more artificial damping is added to the system.

Others have since attempted to improve upon Baraff and Witkin's approach. Desbrun *et al.* [41] precompute the linear part of A, achieving an $O(n)$, unconditionally stable scheme. Their technique, however, is inaccurate and does not generalize well to large systems. Kang *et al.* [98] improve upon this approximation, but ultimately are only replacing the Conjugate Gradient iterations of [72] with a single, Jacobian-like iteration. Volino and Magnenat-Thalmann [99] use a weighted implicit-midpoint method that appeared to give attractive results, but which is less stable and may be difficult to tune in practice. Choi and Ko [95] use the more accurate second-order backward difference formula (BDF2); moreover, their coupled compression/bending model improves the stability of the system, eliminating the need for fictitious spring damping.

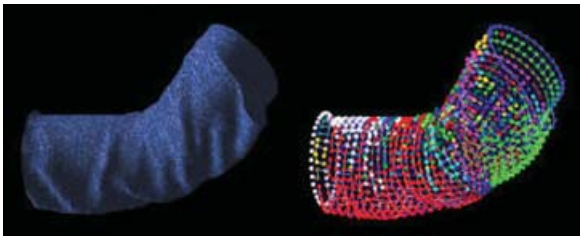

Figure 8: *Decomposing cloth into parts that can be solved more efficiently [101].*

Instead of applying an implicit scheme to the entire system, a number of researchers have employed implicit-explicit (IMEX) schemes [100]. The essential idea is to separately treat the stiff and nonstiff parts of the ODE, handling the stiff parts with an implicit method and the nonstiff parts with an explicit method. This combines the stability of an implicit scheme where needed with the simplicity and efficiency of an explicit scheme where possible. Hauth *et al.* [33] base their IMEX splitting on connection type: stretch springs are handled implicitly, whereas shear and bend "springs" are handled explicitly. This simplifies implementation and improves the sparsity of A—and thus the cost of solving Equation 33. The authors also use BDF2, and embed their linear solver within a Newton solver, making theirs a fully (as opposed to linear) implicit technique. Boxerman and Ascher [101] take a similar approach, optimizing their IMEX splitting adaptively based on a local stability criterion. Each spring is handled either explicitly or implicitly, based on time step, local mesh resolution and material properties. They then take further advantage of the improved system sparsity to decompose the mesh into components that can be solved more efficiently (Figure 8).

Bridson *et al.* [87] apply a similar IMEX splitting to cloth as that used for advection–diffusion equations [100], applying an implicit method to the damping term and an explicit method to the elastic term. This is appropriate when damping plays a significant role, as damping imposes quadratic time step restrictions with respect to mesh size (as opposed to a linear one for elastic terms). They incorporate this within a second-order accurate integration scheme, and include a strain limiting step. Coupled with their careful handling of bending and collisions, they produced visually stunning cloth animations.

4. Lagrangian Mesh-free Methods

4.1. Loosely coupled particle systems

Particle systems were developed by Reeves [102] for explosion and subsequent expanding fire simulation in the movie

"Star Trek II: The Wrath of Khan." The same technique can also be used for modeling other fuzzy objects such as clouds and water, that is, objects that do not have a well-defined surface. Particles are usually graphical primitives such as points or spheres, however, they might also represent complex group dynamics such as a herd of animals [103]. Each particle stores a set of attributes, for example, position, velocity, temperature, shape, age and lifetime. These attributes define the dynamical behavior of the particles over time and are subject to change due to procedural stochastic processes. Particles pass through three different phases during their lifetime: generation, dynamics and death.

In [102], particles are points in 3D space, which represent the volume of an object. A stochastic process generates particles in a predetermined *generation shape* that defines a region about its origin into which the new particles are randomly placed. Attribute values are either fixed or may be determined stochastically. Initially, particles move outward away from the origin with a random speed. During the dynamics phase, particle attributes might change as functions of both time and attributes of other particles. Position is updated by simply adding the velocity. Finally, a particle dies if its lifetime reaches zero or if it does not contribute to the animation anymore, for example, if it is outside of a region of interest. A particle is rendered as a point light source that adds an amount of light depending on its color and transparency attribute. Motion blurring is very easy to achieve by rendering a particle as a streak using its position and velocity. An advantage of particles is their simplicity, which enables the animation of a huge number of particles for complex scenes. The procedural definition of the model and its stochastic control simplifies the human design of the system. Furthermore, with particle hierarchies, complicated fuzzy objects such as clouds can be assembled and controlled. Although the particles are simulated omitting inter-particle forces, the resulting animations are convincing and fast for inelastic phenomena. Therefore, the technique has been widely employed in movies and video games. Examples of modeling waterfalls, ship wakes, breaking waves and splashes using particle systems can be found in [104–108].

Particles that interact with each other depending on their spatial relationship are referred to as spatially coupled particle systems [109]. The interactions between particles evolve dynamically over time, thus, complex geometry and topological changes can be easily modeled using this approach. Tonnesen presented a framework for physically based animation of solids and liquids based on dynamically coupled particles that represent the volume of an object [110,111]. Each particle p_i has a potential energy ϕ_i which is the sum of the pairwise potential energies ϕ_{ij} between p_i and all other other particles p_j, that is,

$$\phi_i = \sum_{j \neq i} \phi_{ij} \tag{36}$$

The force $\mathbf{f}_i$ exerted on p_i with position $\mathbf{x}_i$ is then

$$\mathbf{f}_i = -\nabla_{\mathbf{x}_i} \phi_i = -\sum_{j \neq i} \nabla_{\mathbf{x}_i} \phi_{ij} \tag{37}$$

So far, all particles interact with each other, resulting in $O(n^2)$ complexity where n is the number of particles. The computational costs for computing the force can be reduced to $O(n)$ when restricting the interaction to a neighborhood within a certain distance, and $O(n \log n)$ for neighbor searching. To avoid discontinuities at the neighborhood boundary, the potentials are weighted with a continuous, smooth and monotonically decreasing weighting function that depends on the distance to the particles and ranges from one at the particle position to zero at the neighborhood boundary.

For deriving inter-particle forces, the Lennard–Jones potential ϕ_{LJ} is used

$$\phi_{LJ}(d) = \frac{B}{d^n} - \frac{A}{d^m} \tag{38}$$

where d is the distance between two particles, and n, m, B and A are constants. The Lennard–Jones potential is well known in molecular dynamics for modeling the interaction potential between pairs of atoms. It creates long-range attractive and short-range repulsive forces, yielding particles arranged into hexagonally ordered 2D layers in absence of external forces. A more convenient formulation, called the Lennard–Jones bi-reciprocal function, is written as

$$\phi(d) = \frac{-e_o}{m-n}\left(m\left(\frac{d_0}{d}\right)^n - n\left(\frac{d_0}{d}\right)^m\right) \tag{39}$$

where d_0 is the *equilibrium separation distance*, and $-e_0$ is the minimal potential (the magnitude is called the *dissociation energy*). Increasing the dissociation energy increases the stiffness of the model, while with the exponents n and m the width of the potential well can be varied. Therefore, large dissociation energy and high exponents yield rigid and brittle material, while low dissociation energy and small exponents result in soft and elastic behavior of the object. This allows the modeling of a wide variety of physical properties ranging from stiff to fluid-like behavior. By coupling the dissociation energy with thermal energy such that the total system energy is conserved, objects can be melted and frozen. Furthermore, thermal expansion and contraction can be simulated by adapting the equilibrium separation distance d_0 to the temperature.

One problem of particle systems is that the surface is not explicitly defined. Blinn [112] proposed using an algorithm that was developed to model electron density maps of molecular structures. A Gaussian potential $\varphi_i(\mathbf{x}) = be^{-ad^2}$ (which is often called a *blob*) is assigned to each particle, where a and b are constants and $d = \|\mathbf{x} - \mathbf{x}_i\|$ is the distance from an arbitrary point $\mathbf{x}$ in space to the particle's position $\mathbf{x}_i$. A continuously defined potential field $\varphi(\mathbf{x})$ in space is obtained

by summing the contribution from each particle

$$\varphi(\mathbf{x}) = \sum_i \varphi_i(\mathbf{x}) \tag{40}$$

The surface is then defined as an iso-value S of φ. This yields an implicit coating of the particle that handles topological changes such as splitting and merging by construction. For a more intuitive control of the surface, the constants a and b can be computed as $a = -B/R^2$ and $b = Se^{-B}$, where R is the radius in isolation and B the *blobiness*.

The implicit coating of particles for soft inelastic substances undergoing topological changes poses challenging problems such as volume preservation, avoiding unwanted blending and contact modeling. These were addressed by Desbrun and Cani in a series of papers [113–115]. A hybrid model is used which is composed of two layers: Particles are used to simulate soft inelastic substances as described above, while an elastic implicit layer defines the surface of an object and is locally deformed during collisions. A problem of the implicit coating is that the volume may change significantly during deformation, especially for splitting and merging. However, efficiently computing the volume of a soft substance is not trivial. A *territory* of a particle p_i is defined as the (volumetric) part V_i of the object where the field contribution of p_i is the highest. Note that territories form a partition of the implicit volume of an object. Each particle samples its territory boundary by sending a fixed number of points called *seeds* in a set of distributed directions until they reach the boundary. The volume of a particle is approximated by simply summing up the distances from the particle to the seeds. The local volume variation can then be easily approximated for each particle, and the field function is changed accordingly such that the volume is preserved. Another problem is that split object parts might blend with each other when they come close. To avoid this, an *influence graph* is built at each animation step by recursively adding the neighbors of a particle which are in its sphere of influence to the same influence connected component. Only the particles of the same component can interact and their field functions are blended. However, the problem arises that two or more separated components might collide. For detecting a collision, the seed points on the iso-surface S are tested against the field function of another component. For resolving interpenetrations between two components with potential functions φ_1 and φ_2, exact contact surfaces are computed by applying negative compressing potentials $g_{2,1}$ and $g_{1,2}$ such that $\varphi_1 + g_{2,1} = \varphi_2 + g_{1,2} = S$, resulting in a local compression. To compensate the compression and ensure C^1 continuity of the deformed surfaces, positive dilating potentials are applied in areas defined around the interpenetration zone [113,116]. For collision response, the compression potentials $g_{i,j}$ are computed for each colliding seed and then transmitted as response force to the corresponding particle. Additionally, the two components might be merged locally where the collision force exceeds a threshold.

Szeliski and Tonnesen introduced oriented particles for deformable surface modeling [117,111], where each particle represents a small surface element (called surfel). Each surfel has a local coordinate frame, given by the position of the particle, a normal vector and a local tangent plane to the surface. To arrange the particles into surface-like arrangements, interaction potentials are defined which favor locally planar or locally spherical arrangements. The Lennard–Jones potential described above is used to control the average inter-particle spacing. The weighted sum of all potentials yields the energy of a particle, where the weights control the bending and stiffness of the surface. Variation of the particle energy with respect to its position and orientation yields forces acting on the positions and torques acting on the orientations, respectively. Using these forces and torques, the Newtonian equations for linear and angular motion are solved using explicit integration as described in Section 2.2.

Recently, Bell *et al.* presented a method for simulating granular materials, such as sand and grains, using a particle system [118]. A (nonspherical) grain is composed of several round particles, which move together as a single rigid body. Therefore, stick-slip behavior naturally occurs. Molecular dynamics (MD) is used to compute contact forces for overlapping particles. The same contact model is used for collision of granular materials with rigid bodies, or even between rigid bodies, by simply sampling the rigid body surface with particles.

4.2. Smoothed particle hydrodynamics

Smoothed particle hydrodynamics (SPH) is a Lagrangian technique where discrete, smoothed particles are used to compute approximate values of needed physical quantities and their spatial derivatives. Forces can be easily derived directly from the state equations. Furthermore, as a particle-based Lagrangian approach it has the advantage that mass is trivially conserved and convection is dispensable. This reduces both the programming and computational complexity and is therefore suitable for interactive applications. A drawback of SPH is that it is not possible so far to exactly maintain the incompressibility of material.

Following, we will give a short introduction of the SPH method and discuss applications in computer graphics. For a more detailed introduction we refer the reader to the excellent paper of Monaghan [119].

A function A is interpolated at a position $\mathbf{x}$ from its neighboring particles using a summation interpolant

$$A(\mathbf{x}) = \sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{r}, h) \tag{41}$$

where A_j denotes the function value of a particle p_j at $\mathbf{x}_j$ (the interpolation point), m_j and ρ_j are the mass and density of a particle p_j, respectively and $\mathbf{r} = \mathbf{x} - \mathbf{x_j}$. $W(\mathbf{r}, h)$ is a

smoothing kernel with the properties $\int W(\mathbf{r}, h)d\mathbf{r} = 1$ and $\lim_{h\to 0} W(\mathbf{x}, h) = \delta(\mathbf{x})$, where h is the support radius and δ the Dirac function. For efficiency reasons, h is usually chosen such that the kernel $W(\mathbf{r}, h)$ has compact support. The choice of the smoothing kernel $W(\mathbf{r}, h)$ is very important. Often spline Gaussian kernels are used, see for example, [119] for a discussion of kernels. If $W(\mathbf{r}, h)$ is differentiable, a differentiable interpolant of a function can be derived by simply computing the gradient of $W(\mathbf{r}, h)$, that is,

$$\nabla A(\mathbf{x}) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(\mathbf{r}, h) \tag{42}$$

Therefore, there is no need for finite differences or a grid. To obtain higher accuracy the interpolant can be computed as

$$\rho \nabla A(\mathbf{x}) = \nabla(\rho A(\mathbf{x})) - A(\mathbf{x})\nabla\rho \tag{43}$$

The density is estimated at an arbitrary point as

$$\rho(\mathbf{x}) = \sum_j m_j W(\mathbf{r}, h) \tag{44}$$

The particle densities could be computed for each time step. In practice, this is often not appropriate because the density drops close to the object boundary. Furthermore, difficulties arise when adapting the spatial resolution of the particles [120]. Instead, we can assign an initial density to each particle. The continuity equation (conservation of mass) is then used for computing the variation of density over time

$$\dot{\rho}_i = -\rho_i \nabla \mathbf{v}_i \tag{45}$$

where the divergence of velocity $\nabla \mathbf{v}_i$ is approximated by

$$\nabla \mathbf{v}_i = \frac{1}{\rho_i} \sum_{j\neq i} m_j (\mathbf{v}_j - \mathbf{v}_i) \cdot \nabla W(\mathbf{r}, h) \tag{46}$$

The disadvantage of this update of the density relative to the motion of the neighboring particles is that exact mass conservation is not guaranteed.

The equations of motion are solved by deriving forces. for example, the pressure force can be estimated using Equation 43

$$\mathbf{f}_i^{\text{pressure}} = -m_i \sum_j m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla W(\mathbf{r}, h) \tag{47}$$

where the force is symmetrized to fulfill the action–reaction principle of Newton's third law. For keeping a constant rest density ρ_0, the pressure P_i is computed by a variation of the ideal gas state equation [121]

$$P_i = k(\rho - \rho_0) \tag{48}$$

where k is a gas constant.

For smoothed particles, local stability criteria have been found which, in conjunction with time-adaptive integration, yield both stable and efficient animations. The Courant–Friedrichs–Lewy (CFL) criterion requires that each particle i must not be passed by, giving a limit for the time step $\Delta t \leq \lambda h/c$, where λ is the Courant number and c is the speed of sound of the material, which is the maximum velocity of a deformation wave inside the material. Pressure waves propagate at speed $c = \sqrt{\partial P/\partial\rho}$, therefore using Equation (48) induces $c = \sqrt{k}$. Considering viscosity further decreases the maximum time step, see [119] and [121] for details.

Generally, the animation quality and accuracy increases with a higher number of particles. In [120], an adaptive framework is proposed where space and time resolutions are chosen automatically. Individual particles are refined where pressure differences are above a user-defined threshold. A refined particle p_i with mass m_i and volume m_i/ρ_0 is replaced by n particles with smaller volume and mass m_i/n such that they occupy the same volume as p_i. Considering the particles as spheres with radius proportional to their support radius, an individual support radius h_i can be computed as $h_i = \xi\sqrt{[3]}m_i/\rho_0$, where the constant ξ is chosen according to the desired average number of neighboring particles. Neighboring particles can be merged in stable areas where pressure is almost uniform. However, since a particle is isotropic, the neighboring particles should approximately fill a sphere. The new particle is positioned at the center of gravity of the replaced particles. Its mass and velocity are computed such that mass and linear momentum are conserved.

Considering the object as a set of smeared-out masses, the surface can be defined as an iso-surface of the mass density function [121]. This yields an implicit representation coherent with the physical model. Desbrun and Cani [122] improve this implicit coating of particles by using an active surface that evolves depending on a velocity field, similar to snakes [123]. Therefore, surface tension and other characteristics such as constant volume can be added to the surface model.

4.2.1. *Applications*

SPH was introduced independently by Gingold and Monaghan [124] and Lucy [125], for the simulation of astrophysical problems such as fission of stars. Stam and Fiume introduced SPH to the computer graphics community to depict fire and other gaseous phenomena [126]. They solve the advection–diffusion equation for densities composed of "warped blobs." Desbrun and Cani solve the state equations for the animation of highly deformable bodies using SPH [121]. They achieve the animation of inelastic bodies with a wide range of stiffness and viscosity. Lava flows are animated by coupling viscosity with a temperature field and simulated heat transfer between the particles [127]. By considering hair as a fluid-like continuum, Hadap and Magnenat-Thalmann [128] used a modified formulation of SPH to simulate hair–hair interactions. Premože *et al.* introduced the use of the moving particle semi-implicit method (MPS) for simulating

Figure 9: *Simulating interface tension forces between fluids of different polarity, temperature diffusion and buoyancy [133].*

incompressible multiphase fluids [129]. Impressive visual results were produced by coupling the physical particles with level sets for surface reconstruction. Müller *et al.* presented a method based on SPH and new smoothing kernels, with which fluids with free surfaces can be simulated at interactive rates with up to 5,000 particles [130]. Furthermore, they proposed a term to model surface tension effects. In [131], the interaction of Lagrangian fluids and mesh-based deformable solids is modeled by placing virtual boundary particles, so-called ghost particles [132], on the surface of the solid objects according to Gaussian quadrature. Recently, the SPH method was extended in [133] so that the simulation of phenomena such as boiling water, trapped air and the dynamics of a lava lamp are possible (Figure 9). Liquids with different polarities are simulated by computing a body force that acts perpendicular to the interface of two liquids. The force is proportional to the curvature of the interface and the interface tension. Additionally, air particles are generated and deleted dynamically where air pockets are likely to be formed, making it possible to simulate trapped air.

4.3. Mesh-free methods

Interesting and fairly new approaches to deformable modeling are the so-called *mesh free methods for the solution of partial differential equations*, which originated in the FEM community approximately a decade ago ([134], [135]). For an extensive and up-to-date classification and overview of mesh-free methods, see [136], [137] and also the excellent review paper [138]. A nice introduction to the element-free Galerkin method is given in [139].

4.3.1. *Point-based animations*

Recently, the combination of mesh-free physics with point-sampled surfaces [140] in so-called point-based animations [141] has become popular. Müller *et al.* introduced to the graphics community a mesh-free continuum mechanics-based framework for the animation of elastic, plastic and melting objects [142]. Elastic body forces are derived via the strain energy density

$$U = \frac{1}{2}(\varepsilon \cdot \sigma) \tag{49}$$

where ε is the strain and σ the stress tensor as described in Section 3.1. The elastic force per unit volume at a particle p_i with material coordinates $\mathbf{m}_i$ is computed using the directional derivative $\Delta\nabla_{\mathbf{u}_i}$. For a Hookean material, that is, $\sigma = E \cdot \varepsilon$ (see Equation 4), this yields

$$\mathbf{f}_i = -\nabla_{\mathbf{u}_i} U = -\frac{1}{2}\nabla_{\mathbf{u}_i}(\varepsilon_i \cdot \mathbf{E} \cdot \varepsilon_i) = -\sigma \cdot \nabla_{\mathbf{u}_i}\varepsilon_i \tag{50}$$

where Green's nonlinear strain tensor (see Equation 1) is used for ε_i. For computing ε_i, the spatial derivatives of the displacement field $\nabla\mathbf{u}_i$ at $\mathbf{m}_i$ is needed. To guarantee zero elastic forces for rigid body modes, the approximation of $\nabla\mathbf{u}_i$ from the displacement vectors $\mathbf{u}_j$ of the neighboring particles must be at least first-order accurate. Therefore, the Moving least squares method [143] with a linear basis is used, which yields first-order accurate interpolation of point-sampled functions (contrary to the SPH approximation described in Section 4.2, which is not even zeroth-order accurate). For the following, we will only consider the x-component u of the displacement field $\mathbf{u} = (u, v, w)^T$. The basic idea is to approximate a continuous scalar field $u(\mathbf{m})$ using a Taylor approximation. For particles p_j close to a particle p_i we get a first-order approximation $\tilde{u}_j$ of the values u_j as

$$\tilde{u}_j = u_i + \nabla u|_{\mathbf{m}_i} \cdot \mathbf{m}_{ij} \tag{51}$$

where $\mathbf{m}_{ij} = \mathbf{m}_j - \mathbf{m}_i$ and $\nabla u|_{\mathbf{m}_i} = (u_{,x}, u_{,y}, u_{,z})^T$ at $\mathbf{m}_i$. The index after the coma denotes a spatial derivative. The weighted least-squares error e_i of the approximation $\tilde{u}_j$ is given by

$$e_i = \sum_j (\tilde{u}_j - u_j)^2 w_{ij} \tag{52}$$

where w_{ij} is a normalized, continuously defined and monotonically decreasing weighting function. We want to find the unknowns $u_{,x}$, $u_{,y}$ and $u_{,z}$ such that the error e is minimized. Therefore, we set the derivatives of e with respect to $u_{,x}$, $u_{,y}$ and $u_{,z}$ to zero, yielding three equations for the three unknowns

$$\left(\sum_j \mathbf{m}_{ij}\mathbf{m}_{ij}^T w_{ij}\right)\nabla u|_{\mathbf{m}_i} = \sum_j (u_j - u_i)\mathbf{m}_{ij} w_{ij} \tag{53}$$

Finally, we obtain the spatial derivatives of $u(\mathbf{m})$ at $\mathbf{m}_i$ as

$$\nabla u|_{\mathbf{m}_i} = \mathbf{A}^{-1}\left(\sum_j (u_j - u_i)\mathbf{m}_{ij} w_{ij}\right) \tag{54}$$

where the inverse of $A = \sum_j \mathbf{m}_{ij}\mathbf{m}_{ij}^T w_{ij}$ needs to be computed only once per particle and can be used for computing the derivatives of v and w as well.

With this approach, material properties ranging from stiff elastic to highly plastic can be simulated. In [144], the authors describe how to extend this range to viscous materials such as fluids by merging the solid mechanics equation with the Navier–Stokes equations. Viscosity, pressure and surface

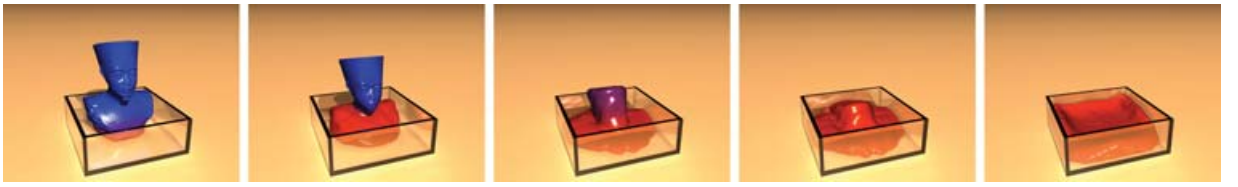

Figure 10: *An elastic solid is dropped onto a heated box and melts to a fluid [144].*

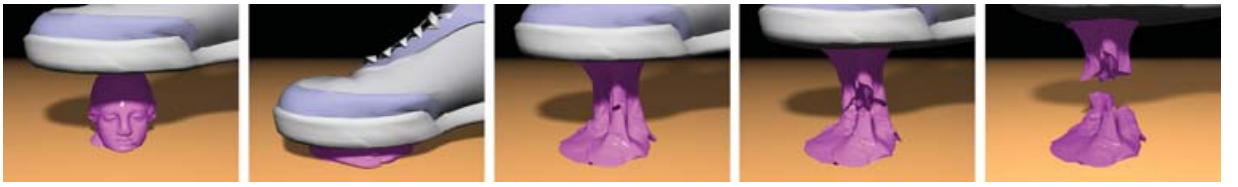

Figure 11: *Highly plastic deformations and ductile fracture [148].*

tension forces are computed using SPH as described in Section 4.2 and added to the elastic force. Note that this allows the simulation of nonrealistic materials such as elastic fluids, similar to [145]. The material properties such as stiffness, viscosity, plasticity and cohesion can be defined per particle. By coupling these properties to the temperature of a particle, local melting and freezing of objects can be achieved (Figure 10).

In the case of elastic materials that do not undergo any topological changes, the surface can be simply dragged along with the particles [142]. This is done by computing the displacement for each surface element (surfel) from the displacements of neighboring particles p_j. The first-order approximation of $\nabla \mathbf{u}$ at $\mathbf{m}_j$ can be used to obtain a smooth displacement vector field that is invariant under linear transformations

$$\mathbf{x}(\mathbf{m}_{s_i}) = \mathbf{m}_{s_i} + \sum_j \overline{\omega}(\mathbf{m}_{s_i}, \mathbf{m}_j)(\mathbf{u}_j + \nabla \mathbf{u}|_{\mathbf{m}_j}(\mathbf{m}_j - \mathbf{m}_{s_i})) \tag{55}$$

where $\mathbf{m}_{s_i}$ and $\mathbf{x}(\mathbf{m}_{s_i})$ are the material and world coordinates of a surfel s_i, respectively, and $\overline{\omega}$ is a normalized weighting kernel. The deformation is computed and applied to both the center and the axes of a surfel. To adapt the sampling of the surface in case of stretching or compression, surfels can be split or merged similar to [146]. The animation of the physics particles combined with this surface skin allows the simulation of elastic and plastic materials of detailed models in real-time. Furthermore, Adams *et al.* [147] showed that the surface can be efficiently raytraced by exploiting the deformation field for bounding hierarchy updates.

However, dragging the surface along with the particles fails if topological changes occur. In this case, an implicit representation can be used as discussed in the Section 4.2. In [142], this implicit representation is sampled with surfels. After an animation step the new position of the surfels is estimated using the displacement computation in Equation (55) The deformed surfels are then projected onto the implicit surface. Finally, a resampling and relaxation scheme ensures that the surface is completely covered and uniformly sampled with surfels. In [144], the implicit representation is improved by applying energy potentials and geometric constraints, which give better control over the surface. However, as the particle representation of the volume is usually much coarser than the surface representation, the implicit coating of the particles cannot represent fine detail of the surface. A possible solution is to use the explicit deformation approach of Equation (55) for solids (with low temperature) the implicit representation for fluids (with high temperature), and blend between the explicit and the implicit representation depending on the temperature for melting object parts. This way, surface detail smoothly fades out and the melting surface approaches the implicit representation. Freezing fluids are handled analogously.

The framework is extended in [148] to cope with fracturing (Figure 11). Cracks are created at surfels where the main principal stress exceeds a threshold. The crack propagates in a plane perpendicular to the main principal stress, where the crack surface is dynamically sampled with surfels. Visibility tests detect neighboring particles, which are separated by a crack surface. A splitting, merging and termination operator handles the topological events that occur when cracks merge or branch. Furthermore, a resampling scheme adapts the particles resolution to handle small fractured object pieces. With the suggested methods a wide range of materials can be fractured, from stiff elastic to highly plastic objects that exhibit brittle and/or ductile fracture.

Collision detection of point-based models has been discussed for rigid [149], quasi-rigid [150] and deformable objects [151]. For rigid collision detection, a bounding sphere hierarchy is built with the surfels as leaves, which is suitable for time-critical collision detection [149]. Pauly *et al.* [150] compute a consistent contact surface and the traction

distribution by solving a linear complementarity problem (LCP). Collision response forces are computed from the local tractions and integration yields the total wrench that acts on the quasi-rigid bodies. Keiser *et al.* propose to decouple the collision handling and deformation to achieve stable and efficient contact handling for deformable objects [151]. Collisions are detected using a high-resolution surface sampled with surfels and a simple scheme is used to compute a consistent contact surface. A penalty force depending on its displacement onto the contact surface is computed for each surfel. These forces are then distributed to the neighboring particles where they are applied as external collision response forces in the next animation step.

Wicke *et al.* propose a method for animating point-sampled thin shells [152]. The geometric surface properties are approximated using splines (so-called *fibers*) embedded in the surface. Physical effects such as plasticity and fracturing can be easily modeled. The authors also describe a skinning technique for point-sampled surfaces, with which a high-resolution surface can be animated using a sparse set of samples.

4.3.2. *Meshless deformations based on shape matching*

Müller *et al.* [34] propose a meshless method for animating deformable objects that does not fit into any of the preceding categories. The nodes of a volumetric mesh are treated as point masses and animated as a simple particle system without connectivity. Then, at every time step, the original configuration of the points (the rest state mesh) is fitted to the actual point cloud in the least squares sense, using shape-matching techniques for point clouds with correspondence. The fitted rest shape yields goal positions for all point masses. Each point mass is then pulled toward its goal position while the displacement divided by the time step is added to the velocity of the point. This geometric integration scheme, although explicit, is shown to be unconditionally stable (Figure 12).

Figure 12: *Stability and the ability to recover from highly deformed or inverted configurations, shown in [34].*

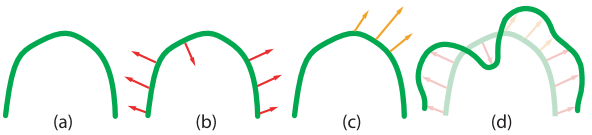

Figure 13: *Reduced deformation models: (a) reference shape* $\mathbf{p}$*, (b) displacement field* $\mathbf{U}_1$*, (c) displacement field* $\mathbf{U}_2$ *and (d) one possible deformed shape* $\mathbf{p}' = \mathbf{p} + \mathbf{U}_1 + 0.5\,\mathbf{U}_2$.

5. Reduced Deformation Models and Modal Analysis

5.1. Basic formulation

As described by James and Pai [153], given N undeformed point locations $\mathbf{p} = [\mathbf{p}_1, \ldots, \mathbf{p}_N]$, a reduced deformation model approximates deformed point locations $\mathbf{p}'$ by a linear superposition of M displacement fields (the columns $\mathbf{U}_i$ of $\mathbf{U}$ in Equation (56). The amplitude of each displacement field is given by the reduced coordinates $\mathbf{q}$ such that

$$\mathbf{p}\prime = \mathbf{p} + \mathbf{U}\mathbf{q} \tag{56}$$

as shown in Figure 13. In [153] it is stated that the columns of $\mathbf{U}$ could arise from any possibly nonlinear black box process, such as an interpolation process, multiresolution modeling or nonlinear/linear modal analysis. The latter will be the focus of this section.

5.2. Linear modal analysis

Pentland and Williams [154] pioneered the use of reduced deformable models in computer graphics, using modal analysis. Given the mass, damping and stiffness matrices $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{K}$, Equation 20 can be decoupled into $N = 3n$ linearly independent (ODEs) by solving the generalized eigenvalue problem

$$\mathbf{M}\Phi\Lambda = \mathbf{K}\Phi \tag{57}$$

such that $\Phi^T\mathbf{M}\Phi = \mathbf{I}$ and $\Phi^T\mathbf{K}\Phi = \Lambda$ are diagonal matrices. The entries of Λ contain the eigenvalues, and the columns of $\Phi = [\Phi_1\ \Phi_2, \ldots, \Phi_N]$ contain the eigenvectors of $\mathbf{M}^{-1}\mathbf{K}$. Φ is often termed as the *modal matrix* or *modal displacement matrix*, where the ith column Φ_i represents the ith mode shape, and the i-th entry of Λ is proportional to the resonant frequency of the Φ_i. The columns of Φ form a basis (the modal basis, or *eigen*basis) of $3n$ dimensional space, so any displacement $\mathbf{u}(t) = \mathbf{x}(t) - \mathbf{x}_0$ can be written as a linear combination of the columns

$$\mathbf{u}(t) = \Phi\mathbf{q}(t) \tag{58}$$

The vector $\mathbf{q}(t)$ contains the modal coordinates (or modal amplitudes). Substituting Equation (58) into Equation (20) and premultiplying by Φ^T yields

$$\Phi^T\mathbf{M}\Phi\,\ddot{\mathbf{q}} + \Phi^T\mathbf{C}\Phi\,\dot{\mathbf{q}} + \Phi^T\mathbf{K}\Phi\,\mathbf{q} = \Phi^T\mathbf{f}_{ext} \tag{59}$$

Note that for general damping, $\Phi^T\mathbf{C}\,\Phi$ is dense, but if we assume proportional (Raleigh) damping $\mathbf{C} = \alpha\mathbf{m} + \beta\mathbf{K}$, then $\Phi^T\mathbf{C}\,\Phi = \alpha\mathbf{I} + \beta\Lambda$ is also diagonal (taking into consideration that $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{K}$ are normally symmetric positive definite). With $\mathbf{g} = \Phi^T\mathbf{f}_{ext}$ we can now express Equation (59) as $3n$

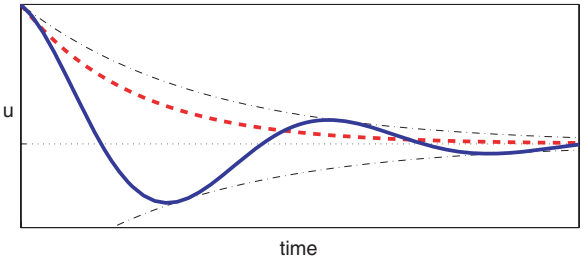

Figure 14: *Underdamped (solid, blue) and critically damped (dashed, red) vibration. The black (dash-point) line is the plot of* $\pm e^{-(\alpha+\beta\lambda_i)t/2}$.

independent scalar second order differential equations

$$\ddot{q}_i + (\alpha + \beta\lambda_i)\dot{q}_i + \lambda_i q_i = g_i \tag{60}$$

where λ_i is the ith diagonal element of Λ.

With this decoupling, one can examine the eigenvalues, discard high frequency mode shapes and thereby only use dominant modes (i.e., dominant columns Φ_i). This can significantly reduce computational cost. Furthermore, the motion components q_i of the individual modes can now be computed independently and combined by linear superposition.

In detail, each of these Equations (60) has an analytical solution of the form

$$q_i = c_1 e^{r_1 t} + c_2 e^{r_2 t} \tag{61}$$

where the constants c_1 and c_2 depend on the initial conditions (q_i and $\dot{q}_i$ at a time $t = t_0$), and the roots of the characteristic equation $r^2 + (\alpha + \beta\lambda_i)r + \lambda_i = 0$ are

$$r_{1,2} = \frac{-(\alpha + \beta\lambda_i) \pm \sqrt{(\alpha + \beta\lambda_i)^2 - 4\lambda_i}}{2} \tag{62}$$

The solution depends on the sign of $R = (\alpha + \beta\,\lambda_i)^2 - 4\lambda_i$: $R > 0$, $R = 0$ and $R < 0$ produces the overdamped (r_1, r_2 real and different), critically damped (r_1, r_2 real but repeated) and underdamped case (r_1, r_2 are complex conjugates), respectively. If r_1, r_2 are real then c_1, c_2 are also real, if r_1, r_2 are complex conjugate pairs then so will be c_1, c_2. In any case, q_i in Equation (61) will be a real value.

Most interest is given to the underdamped case, where damped oscillations occur, and for which Equation (61) can be written as

$$q_i = e^{-(\alpha+\beta\lambda_i)t/2}(c_1\, cos\ \mu t + c_2\, sin\ \mu t) \tag{63}$$

$$\mu = \frac{\sqrt{4\lambda_i - (\alpha + \beta\lambda_i)^2}}{2} \tag{64}$$

see Figure 14. This can be used to compute the response of a mode to an external force at some time $t = t_0$ (see [155]): first, the generalized force $\mathbf{f}_{ext}$ must be transformed to modal coordinates by $\Delta t\mathbf{g} = \Delta t\,\Phi^T\mathbf{f}_{ext}$, and then c_1, c_2 can be computed by substituting Equation (63) into Equation (60) and setting $q_i = 0$ and $\dot{q}_i = \Delta t g$ [155]. Naturally, the system of decoupled ODEs (Equation (60)) can also be numerically integrated through time.

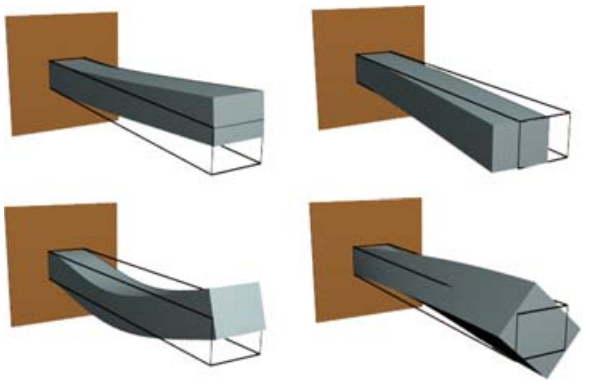

Figure 15: *The first four dominant low frequency mode shapes* $\Phi_1, \ldots, \Phi_4$ *of a constrained bar. The distortion inherent to linear modal analysis is most evident in the bottom right figure [159].*

5.3. Applications and other reduced models

Interestingly, modal analysis was introduced to computer graphics in 1989, but only relatively few significant contributions have been presented since then. The first work of note is Stam's application of modal analysis to the simulation of tree branches subjected to turbulence [156].

James and Pai [157] map runtime dynamics to graphics hardware: in a precomputation stage they build a so-called dynamic response texture (DyRT), where mode shapes Φ_i and other quantities are stored (see Figure 15). At runtime, the modal coordinates $\mathbf{q}$ are computed from rigid bone transforms or external excitations. Finally, displacements are applied to the undeformed mesh by computing $\mathbf{u} = \sum_i \Phi_i q_i$ for a few dominant low frequency modes using a vertex program running on graphics hardware.

Whereas enforcing direct manipulation and collision constraints is relatively straightforward with node positions in Euclidean space, applying these in a modal basis can be somewhat unintuitive. Hauser *et al.* [155] provide a solution, where generalized forces are computed for constrained nodes using the modal basis. Since the force computation involves evaluating a pseudoinverse using the singular value decomposition (SVD), only few constraints (up to ten in their examples) can be applied in a real-time simulation environment. Furthermore, they couple the deformable model with a rigid body reference frame and simulate dynamics, collisions and friction.

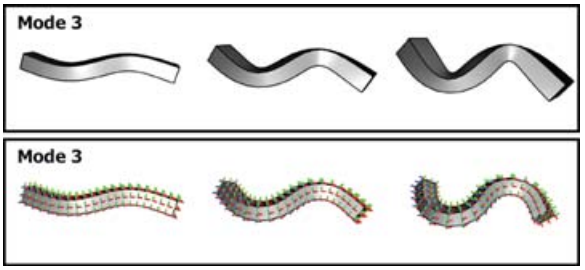

Figure 16: *Evolution of one mode shape in linear modal analysis (top row) and modal warping (bottom row). Image courtesy of Min Gyu Choi, Kwangwoon University.*

In the above methods, a linear Cauchy strain model is employed to obtain the stiffness matrix K, resulting in well-known artifacts for large rotational deformations away from the rest shape (see Section 3.1 and Figure 15). To suppress these artifacts, Choi and Ko [158] identify per-node rotations and extend the basic modal analysis formulation (Section 3.1) to accommodate these rotations, similar in spirit to the warped stiffness approach [38]. Although their modal warping procedure is not guaranteed to perform well for large deformations, their results show visually convincing improvements over the standard linear model (Figure 16). The decoupled ODEs are solved using semi-implicit numerical integration, instead of applying the analytical solution of Equation (61).

Barbič and James [159] take a different approach: instead of employing Cauchy's linearized strain (Equation (2)) they use full quadratic Green strain (Equation (1)) throughout the entire computation, thereby necessitating the solution of a nonlinear version of Equation (20) per time step. The otherwise computationally burdening implicit Newmark integration is greatly accelerated by carrying it out in reduced coordinates (also known as *subspace integration*), and thereafter performing the matrix multiplication of Equation (56) in the fragment shader of current graphics hardware. For the generation of an appropriate reduced deformation basis $\mathbf{U}$, which contains sufficient nonlinear deformation, they provide a fully automatic method based on modal derivatives (Figure 17), and also show how user-defined (force) sketches can assist a full unreduced offline static solver in precomputing a suitable $\mathbf{U}$. Note that due to the nonlinear coupling of modes, using independent harmonic oscillators as previously described (Equation (61)) does not suffice.

A somewhat different model reduction strategy is devised by James and Fatahalian [160]: they entirely precompute collision, contact and dynamics (as well as co-parameterized global illumination) of deformable objects by limiting the range of interactions to a finite set of user impulses. The dynamic responses, termed *impulse response functions* (IRFs), each a chronological set of generalized deformation vectors, are dimensionality reduced using principal component analysis (PCA). At runtime, IRFs are blended based on user interaction.

Figure 17: *Using subspaces based on the first few dominant vibration modes and their derivatives is sufficient to describe large nonlinear deformations [159].*

6. Eulerian and Semi-Lagrangian Methods

There are two general points of view we can take when describing an object that we wish to physically simulate. The main distinction of the two view points is how they discretize an object in order to work with it numerically, and how they define the boundary of the object itself. So far, this paper has covered the *Lagrangian* point of view. As Section 2 explains, the Lagrangian point of view describes an object as a set of moving points (material coordinates) that travel around and change position over time; these points carry their material properties with them as they move through the world. Lagrangian techniques are a convenient way to define an object as a connected mesh of points (Section 4) or simply a cloud of points (Section 3). The *Eulerian* point of view, on the other hand, looks at a stationary set of points and calculates how the material properties stored at those stationary grid points change over time. One of the drawbacks of changing to an Eulerian perspective is that the boundary of the object is no longer explicitly defined. This section covers Eulerian techniques as they are used in computer graphics, and focuses heavily on fluids because fluids are often defined in an Eulerian framework. This section will also explore how object boundaries are represented in the Eulerian framework. This section is not meant to give an exhaustive reference for Eulerian techniques; it is meant to give the reader an idea of what types of deformable and fuzzy objects have been simulated in an Eulerian framework.

The Eulerian approach to fluids was popularized by a series of papers by Foster and Metaxas [161], [162], and [163]. Their formulation solves the fluid (Navier–Stokes) equations on a regular voxel grid and uses a finite difference formulation for all the differential equations. To understand the Eulerian framework, and how it differs from a Lagrangian framework, we will look briefly at how the Navier–Stokes equations (equations of motion for a fluid) are formulated and solved for a single time step. We refer the reader to [164] for an in-depth discussion of liquids, and to [165] for a practical implementation of a smoke simulator.

There are two parts to the Navier–Stokes equations

$$\nabla \cdot \mathbf{u} = 0 \tag{65}$$

$$\mathbf{u}_t = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla \cdot (\nu \nabla \mathbf{u}) - \nabla p + \mathbf{f} \tag{66}$$

These two equations represent the conservation of mass and momentum for an incompressible fluid. The vector field $\mathbf{u}_t$ is the time derivative of the fluid velocity. The scalar pressure field is p[1] , and ν is the kinematic viscosity. The vector field $\mathbf{f}$ represents the body force per unit mass (usually just gravity). In this Eulerian formulation the quantities of the fluid are stored in a grid of cells (a regular grid of cubes or voxels if you like). The velocity is stored on the cell faces and the pressure is stored at the center of the cells. This is commonly called a staggered (or MAC) grid. Note that the position, $\mathbf{x}$, of the fluid is not defined; in this Eulerian frame work, the grid positions remain fixed.

Solving Equations (65) and (66) is usually done in several steps (breaking up a PDE into simpler pieces and solving the pieces separately is known as *operator splitting*). The $\mathbf{f}$ term is simply scaled by the time step and added to the current velocity. The advection term, $-(\mathbf{u} \cdot \nabla)\,\mathbf{u}$, is then solved. One popular way to solve the advection term is with the *semi-Lagrangian* technique [166] because it is stable for large time step sizes. The semi-Lagrangian technique continues as follows—to solve for the advection of any quantity through a velocity field (in this case we are solving for the advection of the velocity field through itself), the velocity at the grid point we wish to update is found, and a path is traced from the grid point backward in time with that velocity to a new location. The quantity that we desire to update is then interpolated from the new location and that interpolated value is used as the updated quantity. It is interesting to note that if the time step is scaled so that the path traverses only a single cell, then the semi-Lagrangian technique is a first-order upwind technique. After advection, the viscosity term is solved. If the viscosity is expected to be large then an implicit technique should be used (see Section 2.2). At this point we have what can be called a *best guess velocity*

$$\tilde{\mathbf{u}} = \mathbf{u} + \Delta t[-(\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla \cdot (\nu \nabla \mathbf{u}) + \mathbf{f}] \tag{67}$$

which does not take into account pressure or mass conservation (Equation (65)). The final step, commonly called a *pressure-projection*, is in fact one of the reasons that Eulerian techniques are popular for fluids. The trick of the pressure-projection step is to solve for a pressure such that subtracting that pressure's gradient from the best guess velocity will give a final velocity that conserves mass. So, we know two things: first, the final velocity, $\mathbf{u}^{\text{new}}$, must contain the pressure term which is missing from Equation (67), so we add it back to get

$$\mathbf{u}^{\text{new}} = \tilde{\mathbf{u}} - \Delta t \nabla p \tag{68}$$

[1]In this formulation we assume a constant density and absorb it into the pressure term for simplicity.

Second, we know the final velocity has to conserve mass, so we plug Equation (68) into Equation (65) to get

$$\nabla \cdot \mathbf{u}^{\text{new}} = \nabla \cdot \tilde{\mathbf{u}} - \Delta t \nabla \cdot (\nabla p) = 0 \tag{69}$$

Rearranging Equation (69) gives us the following Poisson equation

$$\Delta t \nabla^2 p = \nabla \cdot \tilde{\mathbf{u}} \tag{70}$$

with which we must solve for p. We then plug p back into Equation (68) to complete the pressure projection and find our final velocity. The final velocity is incompressible (divergence free), and for a constant density fluid this ensures mass conservation. Compressible fluids can also conserve mass, but their density must change to do so (see [167]).

The system of equations formed by Equation (70) is symmetric and positive definite (as long as there is at least one known Dirichlet boundary condition), and can be solved with a conjugate gradient method. Although the above explanation assumes a regular grid, it can also be carried out on an adaptively refined grid, such as an octree [168,169].

In the rest of this section we will explore the numerous types of deformable objects that are modeled within an Eulerian framework. As the techniques are described we will also describe how the standard Navier–Stokes equations are changed in each case, and how the objects are represented.

In the early work of Foster and Metaxas [161] liquid is displayed either as a height field or a collection of massless particles. These massless particles are different from the material coordinate particles used in a Lagrangian simulation because they are passively moved in the velocity field defined by the fluid, and the particles' velocity is interpolated from the grid. Foster and Metaxas render liquid particles as smoothed ellipsoids [163] with an orientation based on the velocity of the particle and a normal computed from the position of the other nearby particles. Foster and Metaxas also define smoke as massless particles [162]: they advect and diffuse an added temperature field on the grid, and use that temperature to define thermal buoyancy in the smoke to create turbulent motion.

Stam [166] uses a scalar density field to define quantities of smoke. These smoke densities are diffused on the grid and advected with the semi-Lagrangian technique (which Stam did not know about at the time, so when he published his seminal work he called the semi-Lagrangian technique the *method of characteristics*). Stam also adds buoyancy forces based on the local smoke density, and Fedkiw *et al.* [170] add a *vorticity confinement* term to artificially increase small-scale swirling motion that is lost when using semi-Lagrangian advection on coarse grids. Kim *et al.* [171] used the semi-Lagrangian technique with *back and forth error compensation and correction* to keep both small- and large-scale swirling motion in the fluid while maintaining stability. Their technique greatly

Figure 18: *A silver block catapulting some wooden blocks into an oncoming wall of water [174].*

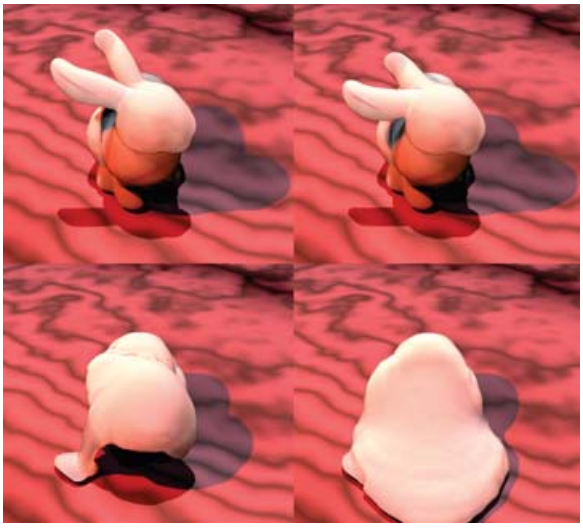

Figure 19: *Melting bunny [172].*

Figure 20: *Bunny made of sand [173]. Image courtesy of Robert Bridson, UBC.*

reduces the dissipation present in the semi-Lagrangian technique, while maintaining its stability.

Carlson *et al.* [172] add a temperature and variable viscosity field to model solid objects that can melt into liquid (Figure 19). In their model, a solid object is simply a fluid with very high viscosity, and as the temperature rises the viscosity decreases, and the object melts into a liquid state. Like [161], the object in their simulation is defined by a set of massless particles, but instead of just rendering the particles they use a splatting technique to extract a meshfrom the particles for rendering. Their method also animates the building of a sand castle, but only high viscosity is used to model the mud.

More recently, Zhu and Bridson [173] model sand as a continuum (fluid) on an Eulerian grid (Figure 20). They use a Lagrangian technique to move particles, then interpolate the velocity from the nearby particles on to the Eulerian grid. They then use the grid velocity to solve the pressure projection step. Thus, their technique combines the strengths of both the Lagrangian and Eulerian techniques. After solving the Navier–Stokes equations, they add in a model for sand. At each grid cell they identify the strain rate (Equation (2)) and use it to calculate stress forces in that cell. All connected groups of cells which can resist the inertia trying to make them slide have their velocity projected to rigid body motion. The velocity in all other cells get forces created by sliding frictional stress added to them. Their method also describes a way to build an implicit function from the particles for rendering.

Carlson *et al.* [174] project rigid bodies on an Eulerian grid to model their interaction with a fluid (Figure 18). A rigid body solver is run in tandem with a fluid solver: at each frame, the rigid bodies that are in contact with fluid are projected onto the fluid, where their equations of motion are solved with a Navier–Stokes equation that is modified with buoyancy and collision forces. After solving the fluid equation they project the velocity in the rigid body cells to rigid body motion (effectively setting the strain rate to zero inside the cells that contain rigid bodies) and use that velocity for the next step of the rigid body solver.

Foster and Fedkiw [175] were the first in graphics to model the boundary of a fluid as a level set. A level set, in this context, is an extra scalar field, ϕ, that stores the signed distance to the fluid surface at each grid point. The change of ϕ is computed in each step with the level set equation,

$$\phi_t + \mathbf{u} \cdot \nabla\phi = 0 \tag{71}$$

which effectively updates the position of the iso-contour that delineates the fluid's surface. The level set method has become one of the most popular ways to define a fluid's surface. It has the drawback of defining the surface only at Eulerian

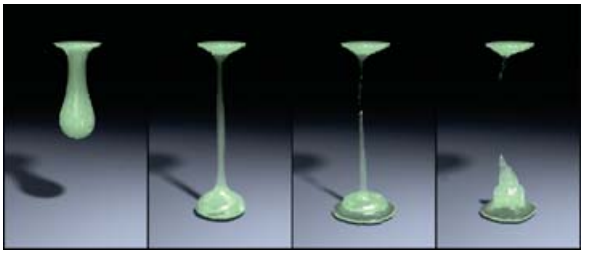

Figure 21: *A dripping viscoelastic fluid [145].*

grid points, and thus rounding off high-resolution details, but this drawback can be reduced by using the particle level set technique [176], which patches the level set values with a group of particles that is passively advected with the surface. The level set can be directly ray traced with a root finder, or a mesh can be extracted from it with marching cubes. We would like to take note of a common confusion when discussing level sets and fluids. Level sets are not used to solve the fluid equations; they are used to define the boundary of the fluid (often called *surface tracking*), so that we know where to solve the fluid equations.

Another way to define a fluid surface is with an explicit polygonal mesh. A mesh representation not only allows a direct way to render the surface, but can also keep track of surface properties like texture coordinates. It can be difficult to deal with topology changes, warping and self intersection in a mesh if a single mesh is used. The contouring method described by Bargteil *et al.* [177] defeats these difficulties by re-defining a mesh for each frame of the fluid simulation. They also maintain a signed distance field and show volume conservation on par with the particle level set method with octree refinement.

Goktekin *et al.* [145] animate viscoelastic substances (that display characteristics of both a liquid and a solid) by adding elastic terms to the basic Navier–Stokes equations (Figure 21). The elastic terms are controlled by von Mises's yield condition and a quasi-linear plasticity model. They keep track of a strain tensor field that is advected throughout the fluid grid, and its components are stored at edge centers (off-diagonal components) and cell centers (diagonal components). The surface of their fluid is defined with a particle level set.

Feldman *et al.* [178] use an Eulerian grid-based technique to model smoke, but they use an unstructured mesh, tetrahedra instead of cubes, near the boundary of complicated objects. Their *hybrid mesh* formulation allows them to model the boundary with curved or highly detailed objects in a more accurate fashion. In their paper they render smoke as a set of massless particles.

7. Conclusion

Physically based deformable models have seen wide application in many fields of computer graphics, and research and development efforts are as active and fruitful as ever. As pointed out throughout this paper, none of the models is suited best for any given application. Instead, many parameters and considerations need to be taken into account, such as model representations, the range of physical parameters, topological changes, real-time or interactive simulation and so on. And even though such a seemingly exhaustive toolkit exists, with which very impressive results have been achieved, we still have quite a way to go until we can plausibly and interactively simulate the natural phenomena in our everyday lives. To quote Richard Feynman [179]

> "The things with which we concern ourselves in science appear in myriad forms, and with a multitude of attributes. For example, if we stand on the shore and look at the sea, we see the water, the waves breaking, the foam, the sloshing motion of the water, the sound, the air, the winds and the clouds, the sun and the blue sky, and light; there is sand and there are rocks of various hardness and permanence, color and texture. There are animals and seaweed, hunger and disease, and the observer on the beach; there may be even happiness and thought. Any other spot in nature has a similar variety of things and influences. It is always as complicated as that, no matter where it is. Curiosity demands that we ask questions, that we try to put things together and try to understand this multitude of aspects as perhaps resulting from the action of a relatively small number of elemental things and forces acting in an infinite variety of combinations."

With the current methodology, the algorithms and models have seen somewhat limited application in production environments and video games. One reason for this is the lack of computational power: many of the presented techniques are inherently offline, and can take hours or days to produce results. In the field of interactive entertainment, small physically based deformable models are already being implemented, and thanks to developments such as the upcoming physics processing unit (PPU), this trend is likely to carry on. A second reason we see for a reluctancy to adopt these new methods is the limited control the users have over the resulting animations. Although this has been previously addressed, we see great potential in coupling more intuitive user interfaces with physically based simulations.

Acknowledgements

We would like to thank the following people for their contributions, suggestions and general help in shaping and improving this paper: Bart Adams, Adam Bargteil, Robert Bridson, Min Gyu Choi, Nico Galoppo and Doug James.

References

1. J. Lasseter. Principles of traditional animation applied to 3D computer animation. In *SIGGRAPH '87*, pp. 35–44, 1987.

2. D. Terzopoulos, J. Platt, A. Barr and K. Fleischer. Elastically deformable models. In *SIGGRAPH '87*, pp. 205–214, 1987.

3. S. F. Gibson and B. Mirtich. *A Survey of Deformable Models in computer Graphics*. Technical Report TR-97-19, MERL, Cambridge, MA, 1997.

4. T. W. Sederberg, J. Zheng, A. Bakenov and A. Nasri. T-splines and T-nurccs. *ACM Transactions on Computer Graphics*, 22(3):477–484, 2003.

5. T. W. Sederberg, D. L. Cardon, G. T. Finnigan, N. S. North, J. Zheng and T. Lyche. T-spline simplification and local refinement. *ACM Transactions on Computer Graphics*, 23(3):276–283, 2004.

6. T. Milliron, R. J. Jensen, R. Barzel and A. Finkelstein. A framework for geometric warps and deformations. *ACM Transactions on Graphics*, 21(1):20–51, 2002.

7. I. Llamas, B. Kim, J. Gargus, J. Rossignac and C. D. Shaw. Twister: A space-warp operator for the two-handed editing of 3D shapes. *ACM Transactions on Graphics*, 22(3):663–668, 2003.

8. M. Botsch and L. Kobbelt. Real-time shape editing using radial basis functions. In *Proceedings of Eurographics, 2005*, pp. 611–621, 2005.

9. O. Sorkine, Y. Lipman, D. Cohen-or, M. Alexa, C. Rössl, H.-P. Seidel. Laplacian surface editing. In *Proceedings of the Eurographics/ ACM SIGGRAPH Symposium on Geometry processing*, pp. 179–188, 2004.

10. Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, H.-Y. Shum. Mesh editing with Poisson-based gradient field manipulation. *ACM Transactions on Computer Graphics*, 23(3):644–651, 2004.

11. M. Botsch and L. Kobbelt. An intuitive framework for real-time freeform modeling. *ACM Transactions on Graphics*, 23(3):630–634, 2004.

12. Y. Lipman, O. Sorkine, D. Levin and D. Cohen-OR. Linear rotation invariant coordinates for meshes. *ACM Transactions on Graphics (ACM SIGGRAPH 2005)* 24(3):479–487, 2005.

13. A. Nealen, O. Sorkine, M. Alexa and D. Cohen-OR. A sketch-based interface for detail-preserving mesh editing. *ACM Transactions on Computer Graphics (ACM SIGGRAPH 2005)*, 24(3):1142–1147, 2005.

14. T. Igarashi, T. Moscovich and J. F. Hughes. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics (ACM SIGGRAPH 2005)* 24(3):1134–1141, 2005.

15. X. C. Wang and C. Phillips. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 129–138, 2002.

16. P. G. Kry, D. L. James and D. K. Pai. Eigenskin: Real time large deformation character skinning in hardware. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 153–159, 2002.

17. D. L. James and C. D. Twigg. Skinning mesh animations. *ACM Transactions on Graphics (SIGGRAPH 2005)*, 24(3):399–407, 2005.

18. A. Love. *A Treatise on the Mathematical Theory of Elasticity*. Cambridge University Press, 1927.

19. A. Witkin and D. Baraff. Physically based modeling: Principles and practice. *SIGGRAPH Course Notes* (1995, 1997).

20. T. J. Chung. *Applied Continuum Mechanics*. Cambridge University Press, NY, 1996.

21. R. D. Cook. *Finite Element Modeling for Stress Analysis*. John Wiley & Sons, NY, 1995.

22. J. Bonet and R. D. Wood. *Nonlinear Continuum Mechanics for finite element Analysis*. Cambridge University Press, 1997.

23. E. E. Gdoutos. *Fracture Mechanics*. Kluwer Academic Publishers, Netherlands, 1993.

24. T. Belytschko, W. K. Liu and B. Moran. *Nonlinear Finite Elements for Continua and Structures*. John Wiley & Sons Ltd., 2000.

25. D. House and D. Breen. *Cloth Modeling and Animation*. A. K. Peters, Ltd., 2000.

26. N. Magnenat-Thalmann, F. Cordier, M. Keckeisen, S. Kimmerle, R. Klein and J. Meseth. Simulation of clothes for real-time applications. In *Eurographics 2004, Tutorials 1: Simulation of Clothes for Real-time Applications*, 2004.

27. N. Magnenat-Thalmann, S. Hadap and P. Kalra. State of the art in hair simulation. In *International Workshop on Human Modeling and Animation*, Korea Computer Graphics Society, pp. 3–9, 2000.

28. P. Volino and N. Magnenat-Thalmann. Animating complex hairstyles in real-time. In *VRST*, pp. 41–48, 2004.

29. M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani,

F. Faure, N. Magnetatthalmann, W. Strasser and P. Volino. Collision detection for deformable objects. *Computer Graphics Forum*, 24(1):61–81, 2005.

30. M. Lin and M. Otaduy. Recent advances in haptic rendering and applications. *SIGGRAPH Course Notes* (2005).

31. W. Press, S. Teukolsky, W. Vetterling and B. Flannery. *Numerical Recipes in C—The Art of Scientific Computing*, 2nd ed. Cambridge University Press, 1992.

32. U. Ascher and L. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial & Applied Mathematics, 1998.

33. M. Hauth, O. Etzmuss and W. Strasser. Analysis of numerical methods for the simulation of deformable models. *The Visual Computer*, 19(7–8):581–600, 2002.

34. M. Müller, B. Heidelberger, M. Teschner and M. Gross. Meshless deformations based on shape matching. *ACM Transactions on Computer Graphics (ACM SIGGRAPH 2005)*, 24(3):471–478, 2005.

35. P. Hunter. *FEM/BEM Notes*. University of Oakland, New Zealand, 2005. http://www.bioeng.auckland.ac.nz/cmiss/fembemnotes/fembemnotes.pdf.

36. J. F. O'Brien and J. K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRAPH* 1999, pp. 287–296, 1999.

37. G. Debunne, M. Desbrun, M.-P. Cani and A. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *Computer Graphics Proceedings*, Annual Conference Series, *ACM SIGGRAPH* 2001, pp. 31–36, 2001.

38. M. Müller, J. Dorsey, L. Mcmillan, R. Jagnow and B. Cutler. Stable real-time deformations. In *Proceedings of the 2002 ACM SIGGRAPH/ Eurographics symposium on Computer animation*, pp. 49–54, 2002.

39. G. Debunne, M. Desbrun, A. Barr and M.-P. Cani. Interactive multiresolution animation of deformable models. In *Eurographics Workshop on Computer Animation and Simulation '99*, pp. 133–144, 1999.

40. G. Debunne, M. Desbrun, M.-P. Cani and A. H. Barr. Adaptive simulation of soft bodies in real-time. In *Computer Animation '00*, pp. 15–20, 2000.

41. M. Desbrun, P. Schröder and A. H. Barr. Interactive animation of structured deformable objects. In *Graphics Interface '99*, pp. 1–8, 1999.

42. J. F. O'Brien, A. W. Bargteil and J. K. Hodgins. Graphical modeling and animation of ductile fracture. In *Proceedings of SIGGRAPH 2002*, pp. 291–294, 2002.

43. J. Smith, A. Witkin and D. Baraff. Fast and controllable simulation of the shattering of brittle objects. In *Graphics Interface*, pp. 81–90, 2000.

44. M. Müller, L. Mcmillan, J. Dorsey and R. Jagnow. Real-time simulation of deformation and fracture of stiff materials. *Proceedings of Eurographics Workshop on Animation and Simulation* 2001, pp. 113–124, $2001.

45. M. Bro-nielsen and S. Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum*, 15(3):57–66, 1996.

46. J.-P. Gourret, N. M. Thalmann and D. Thalmann. Simulation of object and human skin formations in a grasping task. In *SIGGRAPH '89*, pp. 21–30, 1989.

47. S. Cover, N. Ezquerra, J. O'Brien, R. Rowe, T. Gadacz and E. Palm. Interactively deformable models for surgery simulation. *IEEE Computer Graphics and Applications*, 13(6):68–75, 1993.

48. M. A. Sagar, D. Bullivant, G. D. Mallinson and P. J. Hunter. A virtual environment and model of the eye for surgical simulation. In *SIGGRAPH '94*, pp. 205–212, 1994.

49. R. M. Koch, M. H. Gross, F. R. Carls, D. F. Von Büren, G. Fankhauser and Y. I. H. Parish. Simulating facial surgery using finite element models. *Computer Graphics*, 30: 421–428, 1996.

50. S. Cotin, H. Delingette and N. Ayache. Real-time elastic deformations of soft tissues for surgery simulation. In *IEEE Transactions on Visualization and Computer Graphics*, vol. 5(1), pp. 62–73, 1999.

51. S. Cotin, H. Delingette and N. Ayache. A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. *The Visual Computer*, 16(8):437–452, 2000.

52. G. Picinbono, H. Delingette and N. Ayache. Real-time large displacement elasticity for surgery simulation: Non-linear tensor-mass model. *Third International Conference on Medical Robotics, Imaging and Computer Assisted Surgery: MICCAI 2000*, pp. 643–652, 2000.

53. G. Picinbono, H. Delingette and N. Ayache. Nonlinear and anisotropic elastic soft tissue models for medical simulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2001.

54. C. Laugier, C. Mendoza and K. Sundaraj. Towards a realistic medical simulator using virtual environments and force feedback. Injarvis & zelinsky, Editors, *Advanced Robotics*, vol. 6, Springer-Verlag, 2003.

55. M. Müller and M. Gross. Interactive virtual materials. In *GI '04: Proceedings of Graphics Interface 2004*, pp. 239–246, 2004.

56. S. Capell, S. Green, B. Curless, T. Duchamp and Z. Popovic. Interactive skeleton-driven dynamic deformations. In *Proceedings of SIGGRAPH* 2002, pp. 586–593, 2002.

57. X. Wu, M. S. Downes, T. Goktekin and F. Tendick. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. *Eurographics*, 20(3):349–358, 2001.

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

59. E. Grinspun, P. Krysl and P. Schröder. Charms: A simple framework for adaptive simulation. In *Proceedings of SIGGRAPH 2002*, pp. 281–290, 2002.

60. G. Irving, J. Teran and R. Fedkiw. Invertible finite elements for robust simulation of large deformation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 131–140, 2004.

61. M. Müller, M. Teschner and M. Gross. Physically-based simulation of objects represented by surface meshes. In *Proceedings of Computer Graphics International (CGI)*, pp. 26–33, 2004.

62. D. L. James, J. Barbič and C. D. Twigg. Squashing cubes: Automating deformable model construction for graphics. In *Proceedings of the SIGGRAPH 2004 Conference on Sketches* & Applications, 2004.

63. N. Molino, Z. Bao and R. Fedkiw. A virtual node algorithm for changing mesh topology during simulation. *ACM Transactions on Graphics*, 23(3):385–392, 2004.

64. D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *SIGGRAPH* '88, pp. 269–278, 1988.

65. D. Terzopoulos and A. Witkin. Physically based models with rigid and deformable components. *IEEE Computer Graphics and Applications*, 8(6):41–51, 1988.

66. J. Teran, S. Blemker, V. N. T. Hing and R. Fedkiw. Finite volume methods for the simulation of skeletal muscle. In *Eurographics/SIGGRAPH Symposium on Computer Animation*, pp. 68–74, 2003.

67. D. L. James and D. K. Pai. ArtDefo: Accurate real time deformable objects. In *SIGGRAPH '99*, pp. 65–72, 1999.

68. D. L. James and D. K. Pai. Real time simulation of multizone elastokinematic models. In *2002 IEEE International Conference on Robotics and Automation*, pp. 927–932, 2002.

69. D. L. James and D. K. Pai. Multiresolution Green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Transactions on Graphics*, 22(1):47–82, 2003.

70. Y. Chen, Q. Zhu, A. Kaufman and S. Muraki. Physically-based animation of volumetric objects. In *CA '98: Proceedings of the Computer Animation*, p. 154. 1998.

71. D. Breen, D. House and M. Wozny. Predicting the drape of woven cloth using interacting particles. In *SIGGRAPH '94*, pp. 365–372, 1994.

72. D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proceedings of SIGGRAPH* 1998, pp. 43–54, 1998.

73. S. M. Platt, N. I. Badler. Animating facial expressions. In *SIGGRAPH '81*, pp. 245–252, 1981.

74. K. Waters. A muscle model for animation three-dimensional facial expression. In *SIGGRAPH '87*, pp. 17–24, 1987.

75. J. Chadwick, D. Haumann and R. Parent. Layered construction for deformable animated characters. In *SIGGRAPH '89*, pp. 243–252, 1989.

76. D. Terzopoulos and K. Waters. Physically-based facial modeling, analysis, and animation. *Journal of Visualization and Computer Animation*, 1(1):73–80, 1990.

77. K. Waters and D. Terzopoulos. Modeling and animating faces using scanned data. *Journal of Visualization and Computer Animation*, 2(2):123–128, 1991.

78. G. S. P. Miller. The motion dynamics of snakes and worms. In *SIGGRAPH '88*, pp. 169–173, 1988.

79. X. Tu and D. Terzopoulos. Artificial fishes: physics, locomotion, perception, behavior. In *SIGGRAPH '94*, pp. 43–50, 1994.

80. D. Terzopoulos, J. Platt and K. Fleischer. Heating and melting deformable models (from goop to glop). In *Graphics Interface '89*, pp. 219–226, 1989.

81. S. Kawabata. The standardization and analysis of hand evaluation. *The Textile Machinery Society of Japan*, 1980.

82. O. Etzmuss, M. Keckeisen and W. Strasser. A fast finite element solution for cloth modelling. *Proceedings of Pacific Graphics*, 2003.

83. M. Kass. An introduction to physically based modeling, chapter: Introduction to continuum dynamics for computer graphics. In *SIGGRAPH 95 Course Notes* (1995).

84. O. Etzmuss, J. Gross and W. Strasser. Deriving a particle system from continuum mechanics for the animation of deformable objects. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):538–550, 2003.

85. P. Volino and N. Magnenat-Thalmann. Developing simulation techniques for an interactive clothing system. In *Proceedings of the 1997 International Conference on Virtual Systems and MultiMedia*, p. 109, 1997.

86. D. Bourguignon and M.-P. Cani. Controlling anisotropy in mass-spring systems. In *Computer Animation and Simulation '00*, pp. 113–123, 2000.

87. R. Bridson, S. Marino and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *ACM SIGGRAPH/Eurographics Symposium Computer Animation*, pp. 28–36, 2003.

88. E. Grinspun, A. N. Hirani, M. Desbrun and P. SchrÖDER. Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pp. 62–67, 2003.

89. K. Bhat, C. Twigg, J. K. Hodgins, P. Khosla, Z. Popovic and S. Seitz. Estimating cloth simulation parameters from video. In *ACM SIGGRAPH/ Eurographics Symposium on Computer Animation*, pp. 37–51, 2003.

90. G. Bianchi, B. Solenthaler, G. Székely and M. Harders. Simultaneous topology and stiffness identification for mass-spring models based on fem reference deformations. In *MICCAI 2* 2004, pp. 293–301, 2004.

91. J. Villard and H. Borouchaki. Adaptive meshing for cloth animation. In *11th International Meshing Roundtable* (Ithaca, New York, USA), Sandia National Laboratories, pp. 243–252, 2002.

92. D. Hutchinson, M. Preston and T. Hewitt. Adaptive refinement for mass/spring simulations. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96*, pp. 31–45, 1996.

93. L. Li and V. Volkov. Cloth animation with adaptively refined meshes. In *ACSC*, pp. 107–114, 2005.

94. B. Eberhardt, A. Weber and W. Strasser. A fast, flexible, particle system model for cloth draping. *IEEE Computer Graphics and Applications*, 16(5):52–59, 1996.

95. K.-J. Choi and H.-S. Ko. Stable but responsive cloth. In *SIGGRAPH '02*, pp. 604–611, 2002.

96. X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In *Proceedings Graphics Interface*, pp. 147–154, 1995.

97. U. Ascher and E. Boxerman. On the modified conjugate gradient method in cloth simulation. *The Visual Computer*, 19(7–8):526–531, 2003.

98. Y. Kang, J. Choi, H. Cho, D. Lee and C. Park. Real-time animation technique for flexible and thin objects. In *WSCG 2000*, pp. 322–329, 2000.

99. P. Volino and N. Magnenat-Thalmann. Implementing fast cloth simulation with collision response. *IEEE Computer Society*, pp. 257–268, 2000.

100. U. Ascher, S. Ruuth and B. Wetton. Implicit-explicit methods for time-dependent PDE's. *SIAM Journal of Numerical Analysis*, 32, pp. 797–823, 1995.

101. E. Boxerman and U. Ascher. Decomposing cloth. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 153–161, 2004.

102. W. T. Reeves. Particle systems—a technique for modeling a class of fuzzy objects. *ACM Transactions on Computer Graphics*, 2(2):91–108, 1983.

103. C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87*, pp. 25–34, 1987.

104. A. Fournier and W. T. Reeves. A simple model of ocean waves. In *SIGGRAPH* '86, pp. 75–84, 1986.

105. D. R. Peachey. Modeling waves and surf. In *SIGGRAPH '86*, pp. 65–74, 1986.

106. M. E. Goss. Motion simulation: A real time particle system for display of ship wakes. *IEEE Computer Graphics and Applications*, 10(3):30–35, 1990.

107. K. Sims. Particle animation and rendering using data parallel computation. In *SIGGRAPH '90*, pp. 405–413, 1990.

108. J. F. O'Brien and J. K. Hodgins. Dynamic simulation of splashing fluids. In *CA '95: Proceedings of the Computer Animation*, p. 198, 1995.

109. D. Tonnesen. Spatially coupled particle systems. 4.1–4.21.

110. D. Tonnesen. Modeling liquids and solids using thermal particles. In *Graphics Interface*, pp. 255–262, 1991.

111. D. Tonnesen. *Dynamically Coupled Particle Systems for Geometric Modeling, Reconstruction, and Animation*. Ph.D. thesis, University of Toronto, November 1998.

112. J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

113. M.-P. Cani. An implicit formulation for precise contact modeling between flexible solids. In *SIGGRAPH '93*, pp. 313–320, 1993.

114. M. Desbrun and M.-P. Highly deformable material for animation and collision processing. In *5th Eurographics Workshop on Animation, Simulation*, 1994.

115. M. Desbrun and M.-P. Cani. Animating soft substances with implicit surfaces. In *Computer Graphics Proceedings*, *ACM SIGGRAPH*, pp. 287–290, 1995.

116. A. Opalach and M. Cani. Local deformations for animation of implicit surfaces. In *13th Spring Conference on Computer Graphics*,W. straβer, editor, pp. 85–92, 1997.

117. R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics*, 26(2):185–194, 1992.

118. N. Bell, Y. Yu and P. J. Mucha. Particle-based simulation of granular materials. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/ Eurographics symposium on Computer animation*, pp. 77–86. 2005.

119. J. Monaghan. Smoothed particle hydrodynamics. *Annual Review of Astronomical Physics*, 30, 543, 1992.

120. M. Desbrun, and M.-P. Cani. *Space-Time Adaptive Simulation of Highly Deformable Substances*. Technical Report, INRIA Nr. 3829, 1999.

121. M. Desbrun and M.-P. Cani. Smoothed particles: A new paradigm for animating highly deformable bodies. In *6th Eurographics Workshop on Computer Animation and Simulation '96*, pp. 61–76, 1996.

122. M. Desbrun and M.-P. Cani. Active implicit surface for animation. In *Proceedings of Graphics Interface*, pp. 143–150, 1998.

123. M. Kass, A. Witkin and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

124. R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics—theory and application to non-spherical stars. *Royal Astronomical Society, Monthly Notices*, 181: 375–389, 1977.

125. L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82(12):1013–1024, 1977.

126. J. Stam and E. Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In *SIGGRAPH '95*, pp. 129–136, 1995.

127. D. Stora, P.-O. Agliati, M.-P. Cani, F. Neyret and J.-D. Gascuel. Animating lava flows. In *Proceedings of Graphics Interface*, pp. 203–210, 1999.

128. S. Hadap and N. Magnenat-Thalmann. Modeling dynamic hair as continuum. In *Eurographics Proceedings. Computer Graphics Forum*, 20(3):329–338, 2001.

129. S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn and R. T. Whitaker. Particle-based simulation of fluids. In *Proceedings of Eurographics 2003*, pp. 401–410, 2003.

130. M. Müller, D. Charypar and M. Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pp. 154–159, 2003.

131. M. Müller, S. Schirm, M. Teschner, B. Heidelberger and M. Gross. Interaction of fluids with deformable solids. In *Computer Animation and Social Agents CASA'04*, pp. 159–171, 2004.

132. J. J. Monaghan. Simulating free surface flows with SPH. *Journal of Computational Physics*, 110(2):399–406, 1994.

133. M. Müller, B. Solenthaler, R. Keiser and M. Gross. Particle based fluid-fluid interaction. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 237–244, 2005.

134. B. Nayroles, G. Touzot, P. Villon and A. Ricard. Generalizing the finite element method: Diffuse

approximation and diffuse elements. *Computational Mechanics*, 10(5):307–318, 1992.

135. N. Sukumar. Meshless methods and partition of unity finite elements. In Proceedings *of the 6th International ESAFORM Conference on Material Forming*, pp. 603–606, 2003.

136. T.-P. Fries and H. G. Matthies. *Classification and Overview of Meshfree Methods*. Technical Report, TU Brunswick, Germany Nr. 2003-03, 2003.

137. G. R. Liu. *Mesh-Free Methods*. CRC Press, 2002.

138. T. Belytschko, Y. Krongauz, D. Organ, M. Fleming and P. Krysl. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering*, 139(3):3–47, 1996.

139. H. Askes. *Everything You Always Wanted to Know About the Element-Free Galerkin Method, and More*. Technical Report, TU Delft nr. 03.21.1.31.29, 1997.

140. H. Pfister, M. Zwicker, J. Van BAAR and M. Gross. Surfels: Surface elements as rendering primitives. In *SIGGRAPH 2000, Computer Graphics Proceedings*, pp. 335–342, 2000.

141. Point based animation: Resource collection on the world-wide web. http://www.pointbasedanimation.org, 2004.

142. M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 141–151, 2004.

143. P. Lancaster and K. Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of Computation*, 87, 141–158, 1981.

144. R. Keiser, B. Adams, D. Gasser, P. Bazzi, P. DutrÉ and M. Gross. A unified Lagrangian approach to solid-fluid animation. In *Proceedings of the Symposium on Point-Based Graphics*, pp. 125–133, 2005.

145. T. G. Goktekin, A. W. Bargteil and J. F. O'Brien. A method for animating viscoelastic fluids. In *Proceedings of ACM SIGGRAPH*, vol. 23, pp. 463–468, 2004.

146. M. Pauly, R. Keiser, L. P. Kobbelt and M. Gross. Shape modeling with point-sampled geometry. In *Proceedings of ACM SIGGRAPH*, pp. 641–650, 2003.

147. B. Adams, R. Keiser, M. Pauly, L. J. Guibas, M. Gross and P. Dutré. Efficient raytracing of deforming point-sampled surfaces. In *Proceedings of Eurographics 2005*, pp. 677–684, 2005.

148. M. Pauly, R. Keiser, B. Adams, P. Dutré, M. Gross and L. J. Guibas. Meshless animation of fracturing solids. *ACM Transactions on Computer Graphics*, 24(3):957–964, 2005.

149. J. Klein and G. Zachmann. Point cloud collision detection. In *Proceedings of EUROGRAPHICS 2004*, pp. 567–576, 2004.

150. M. Pauly, D. K. Pai and L. J. Guibas. Quasi-rigid objects in contact. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 109–119, 2004.

151. R. Keiser, M. Müller, B. Heidelberger, M. Teschner and M. Gross. Contact handling for deformable point-based objects. In *Proceedings of Vision, Modeling, Visualization VMV'04*, pp. 339–347. Nov 2004.

152. M. Wicke, D. Steinemann and M. Gross. Efficient animation of point-based thin shells. In *Proceedings of Eurographics '05*, pp. 667–676, 2005.

153. D. L. James and D. K. Pai. BD-Tree: Output-sensitive collision detection for reduced deformable models. *ACM Transactions on Graphics (SIGGRAPH 2004)*, 23(3):393–398, 2004.

154. A. Pentland and J. Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics (Proceedings of SIGGRAPH* 1989) 23(3):215–222, 1989.

155. K. K. Hauser, C. Shen and J. F. O'Brien. Interactive deformation using modal analysis with constraints. In *Graphics Interface* '03, pp. 247–255, 2003.

156. J. Stam. Stochastic dynamics: Simulating the effects of turbulence on flexible structures. *Computer Graphics Forum*, 16(3):159–164.

157. D. L. James and D. K. Pai. DyRT: Dynamic response textures for real time deformation simulation with graphics hardware. In *Proceedings of SIGGRAPH 2002*, pp. 582–585, 2002.

158. M. G. Choi and H.-S. Ko. Modal warping: Real-time simulation of large rotational deformation and manipulation. *IEEE Transactions on Visualization and Computer Graphics*, 11(1):91–101, 2005.

159. J. Barbič and D. L. James. Real-time subspace integration for St. Venant- Kirchhoff deformable models. *ACM Transactions on Computer Graphics (ACM SIGGRAPH 2005)*, 24(3):982–990, 2005.

160. D. L. James and K. Fatahalian. Precomputing interactive dynamic deformable scenes. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003)*, 22(3):879–887, 2003.

161. N. Foster and D. Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, 1996.

162. N. Foster and D. Metaxas. Modeling the motion of a hot, turbulent gas. In *SIGGRAPH '97*, pp. 181–188, 1997.

163. N. Foster and D. Metaxas. Controlling fluid animation. In *Proceedings CGI '97*, pp. 178–188, 1997.

164. M. T. Carlson. *Rigid, Melting, and Flowing Fluid*. Ph.D. thesis, Georgia Institute of Technology, 2004.

165. J. Stam. Real-time fluid dynamics for games. In *Proceedings of the Game Developer Conference*, 2003.

166. J. Stam. Stable fluids. In *SIGGRAPH '99*, pp. 121–128, 1999.

167. G. D. Yngve, J. F. O'Brien and J. K. Hodgins. Animating explosions. In *Proceedings of SIGGRAPH*, Computer Graphics Proceedings, Annual Conference Series, pp. 29–36, 2000.

168. L. Shi and Y. Yu. *Visual Smoke Simulation with Adaptive Octree Refinement*. Technical Report UIUCDCS-R-2002-2311, University of Illinois at Urbana–Champaign, 2002.

169. F. Losasso, F. Gibou and R. Fedkiw. Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics*, 23(3):457–462, 2004.

170. R. Fedkiw, J. Stam and H. W. Jensen. Visual simulation of smoke. In *SIGGRAPH '01*, pp. 15–22, 2001.

171. B. Kim, Y. Liu, I. Llamas and J. Rossignac. Flowfixer: Using bfecc for fluid simulation. In *Eurographics Workshop on Natural Phenomena*, pp. 51–56, August 2005.

172. M. Carlson, P. Mucha, R. B. Van Horn III and G. Turk. Melting and flowing. In *Proceedings of the 2002 ACM SIGGRAPH Symposium on Computer Animation*, pp. 167–174, 2002.

173. Y. Zhu and R. Bridson. Animating sand as a fluid. *ACM Transactions on Computer Graphics (Proceedings of ACM SIGGRAPH 2005)*, 24(3):965–971, 2005.

174. M. Carlson, P. J. Mucha and G. Turk. Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Transactions on Graphics*, 23(3):377–384, 2004.

175. N. Foster and R. Fedkiw. Practical animation of liquids. In *Proceedings of ACM SIGGRAPH 2001*, pp. 23–30, 2001.

176. D. Enright, S. Marschner and R. Fedkiw. Animation and rendering of complex water surfaces. In *SIGGRAPH '02*, pp. 736–744, 2002.

177. A. W. Bargteil, T. G. Goktekin, J. F. O'Brien and J. A. Strain. A semi-Lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics*, 2005, 25(1):19–38, January 2006.

178. B. E. Feldman, J. F. O'Brien and B. M. Klingner. Animating gases with hybrid meshes. In *Proceedings of ACM SIGGRAPH* 2005, 24(3):904–909, 2005.

179. R. P. Feynman, R. B. Leighton and M. Sands. *The Feynman Lectures on Physics: Volume I*. Addison Wesley, 1963.