Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2007)
D. Metaxas and J. Popovic (Editors)

Hybrid Simulation of Deformable Solids

Eftychios Sifakis† Stanford University Intel Corporation
Tamar Shinar† Stanford University
Geoffrey Irving† Stanford University Pixar Animation Studios
Ronald Fedkiw† Stanford University Industrial Light+Magic

Abstract

Although mesh-based methods are efficient for simulating simple hyperelasticity, maintaining and adapting a mesh-based representation is less appealing in more complex scenarios, e.g. collision, plasticity and fracture. Thus, meshless or point-based methods have enjoyed recent popularity due to their added flexibility in dealing with these situations. Our approach begins with an initial mesh that is either conforming (as generated by one's favorite meshing algorithm) or non-conforming (e.g. a BCC background lattice). We then propose a framework for embedding arbitrary sample points into this initial mesh allowing for the straightforward handling of collisions, plasticity and fracture without the need for complex remeshing. A straightforward consequence of this new framework is the ability to naturally handle T-junctions alleviating the requirement for a manifold initial mesh. The arbitrarily added embedded points are endowed with full simulation capability allowing them to collide, interact with each other, and interact with the parent geometry in the fashion of a particle-centric simulation system. We demonstrate how this formulation facilitates tasks such as arbitrary refinement or resampling for collision processing, the handling of multiple and possibly conflicting constraints (e.g. when cloth is nonphysically pinched between two objects), the straightforward treatment of fracture, and sub-element resolution of elasticity and plasticity.

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

1. Introduction

Simulation of deformable models was pioneered in computer graphics by [TPBF87, TF88a, TF88b]. It has influenced research in cloth animation [BW98], muscle and face simulation [TSSB*05, SNF05], flesh deformation [CGC*02b], virtual surgery [SHGS06] and fracture [OH99].

Mesh-based methods require the generation of an initial simulation mesh, which can be conforming (e.g. [MBTF03, ACSYD05]), or non-conforming when used in conjunction with an embedded simulation technique (e.g. [CGC*02a, CGC*02b, MBF04, MG04, MTG04]) in which case a simple cube or BCC background mesh can be used. Generation of a conforming simulation mesh is typically non-trivial and is best suited to applications that require no changes to the initial mesh. However, if the mesh needs to be adapted on the fly, e.g. for fracture [OH99], remeshing can be prohibitively expensive and can introduce poor quality elements. [MBF04] proposed a partial solution that uses embedded simulation technology to fracture tessellated objects without continuous remeshing, allowing elements to be modeled as partially full of material. Limitations, including the restriction that edges cannot be fractured twice, prevent this method from being completely general. Collision processing can often require more surface resolution than is present in the initial simulation mesh. While some have considered adaptive frameworks [DDCB01, GKS02, CGC*02b], it is wasteful to refine the full volumetric mesh in the vicinity of the boundaries solely for collisions. Although not as efficient as mesh-based methods for the simulation of elastic deformation, point-based methods provide added flexibility making them attractive for applications involving fracture, virtual surgery, resampling for collision handling, etc., see e.g. [MKN*04, PKA*05, SOG06, WSG05, MHTG05].

Starting with either a conforming or non-conforming mesh, we propose a method for embedding an arbitrary point in this mesh. We call this a *hard binding*. To derive the relationships between physical quantities of the embedded point and the mesh, we start by considering a hypothetical refinement of the mesh that resolves the embedded point. We compute internal finite element forces for these hypothetical subelements and illustrate how to redistribute these forces to the parent mesh. The mass of the embedded particle is redistributed to the parent mesh as well. This redistribution is

† email: {sifakis|shinar|irving|fedkiw}@graphics.stanford.edu

Symposium on Computer Animation 2007, San Diego, CA, August 03-04, 2007.
© 2007 ACM 978-1-59593-624-4/07/0008 $5.00

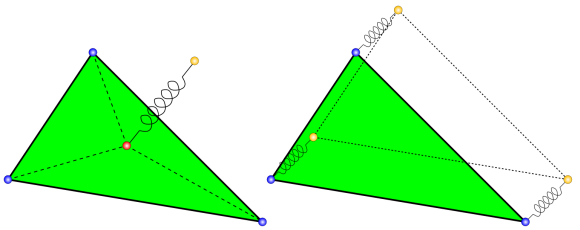

Figure 1: *Parent particles (blue), hard bound target locations (red) and soft bound particles (yellow). Soft bound duplication of a hard bound target location (left), and three soft bound duplications of parent particles (right).*

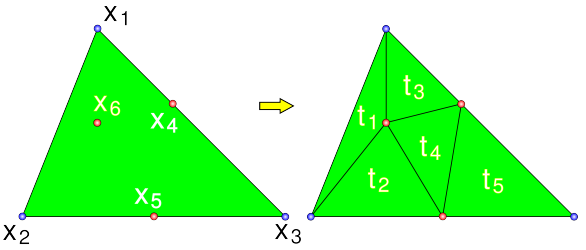

Figure 2: *Resolution of hard bound particles (depicted in red) via refinement of the parent element.*

shown to be independent of the chosen refinement. Notably, this approach results in automatic and natural handling of T-junctions, as masses and forces of T-junction nodes can be redistributed to the parent mesh. This specific handling of T-junctions is similar to that in [LC06]. We then generalize this approach to arbitrary embeddings. Hard bound particles have their mass and any forces or impulses applied to them redistributed to the parents in a natural fashion while maintaining the notion of an *effective mass* so that they can fully participate in various numerical algorithms.

A hard binding constrains an embedded particle to its barycentrically determined location (a similar heuristic was given in [GBT06]). Thus, hard bound particles are not particles at all (i.e. they do not possess any degrees of freedom), but merely target locations that live on the parent mesh. In order to transition to a fully particle-centric simulation framework, we create the notion of a *soft binding*. A soft binding is an abstract connection between a real particle with full degrees of freedom and a hard binding target location enabling full two-way interaction. Soft bound particles are free to interact with each other and the parent mesh constituting a fully particle-centric framework. Soft bindings can be created between any particle and any target location even duplicating the original degrees of freedom in the mesh itself (see figure 1). The soft binding mechanism is responsible for the two-way interaction between the particle-based system and the mesh-based framework. Although one could implement this connection using simple springs, we have designed a more sophisticated soft binding interface which is notably fully implicit allowing one to stiffen the two-way interaction to the point where the soft bound particle always lies on its target location up to numerical precision, without stability issues or additional time step restrictions.

Our work shares the motivation of [GBT06] to incorporate particle constraints in the simulation of deformable objects, and the function of their geometric constraints is conceptually analogous to our hard bindings. However, their approach works by introducing external forces dependent on the integration method used whereas our hard bindings are enforced through the conjugate operations of force distribution and velocity interpolation. That is, instead of forcing our constrained degrees of freedom toward their target locations, we directly project them out of the equations of motion. This allows seamless incorporation of such constraints into any time integration scheme, including globally coupled implicit schemes. Another major difference is that our framework allows optional drift of particles away from their target locations through soft bindings.

Novel contributions of our work include the tight integration of hard binding constraints and unconditionally stable binding spring forces into the semi-implicit Newmark time integration scheme, lag-free duplication of degrees of freedom through soft bindings by force transfer from parent particles, and integration of rigid/deformable coupling into the Newmark scheme and conjugate gradient solver. We present these contributions as part of a broad hybrid simulation framework building on simple, physically motivated principles. We demonstrate the features of this framework with examples that include dynamically adapting the surface sampling density for collisions, duplicating parent mesh particles to resolve conflicting constraints caused by the non-physical pinching of cloth, the facilitation of fracture and cutting algorithms, and an extension of our framework to two-way interactions with rigid bodies.

2. Hard Bindings

The simulation mesh is subject to internal forces, from for example finite elements, as well as external forces from gravity, friction, collisions, etc., and we want these forces to act on the hard bound particles as well. We motivate our approach for propagating forces to the parent mesh by considering a refinement that resolves all the hard bound particles. Figure 2 shows three hard bound particles along with an associated refinement. Although this refinement is not unique, it turns out that the propagation of physical properties from the hard bound particles to the parent mesh is independent of the tessellation used and can be performed without explicitly refining the parent element. Internal forces are defined on the subelements in standard fashion, but must be remapped to the parent particles since the hard bound particles are not free to move independently.

Conservative forces can be defined in terms of the gradient $\mathbf{f} = -\partial\Psi(\mathbf{x})/\partial\mathbf{x}$ of the potential energy $\Psi(\mathbf{x})$. The potential energy of the parent triangle is $\Psi = \sum\Psi_i$, where Ψ_i

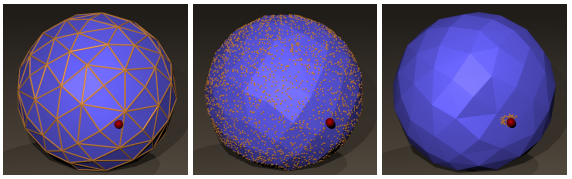

Figure 3: *(Left) A coarse deformable ball is penetrated by a kinematic sphere. (Center) Sprinkling hard bound particles on the surface of the sphere allows one to resolve the collisions without requiring refinement of the tetrahedral simulation mesh. (Right) Hard bound particles can be dynamically added and removed based on proximity to collision objects.*

is the potential energy of subtriangle t_i. Each Ψ_i is naturally defined in terms of the positions of the vertices of triangle t_i rather than the positions of the parent particles, although the former are fully determined by the latter through the binding constraint. Let $\mathbf{x}_i$ denote the positions of the vertices of triangle t_i, e.g. $\mathbf{x}_2$ is composed of the positions of particles $\{x_2, x_5, x_6\}$ in figure 2. The parent triangle vertices $\{x_1, x_2, x_3\}$ are represented by $\mathbf{x}$. Using this notation, the three forces on the parent particles are

$$\mathbf{f} = -\sum_i \frac{\partial \Psi_i}{\partial \mathbf{x}} = -\sum_i \left(\frac{\partial \mathbf{x}_i}{\partial \mathbf{x}}\right)^T \frac{\partial \Psi_i}{\partial \mathbf{x}_i} = \sum_i \left(\frac{\partial \mathbf{x}_i}{\partial \mathbf{x}}\right)^T \mathbf{f}_i \quad (1)$$

where $\mathbf{f}_i = -\partial \Psi_i / \partial \mathbf{x}_i$ are the three vertex forces computed on each subtriangle t_i. The Jacobian $\partial \mathbf{x}_i / \partial \mathbf{x}$ is constant for each triangle t_i and contains the barycentric coordinates of $\mathbf{x}_i$ with respect to the vertices $\mathbf{x}$ of the parent triangle. Equation (1) can be extended to the computation of elastic force differentials for implicit or quasistatic time integration via

$$\delta \mathbf{f}|_{\delta \mathbf{x}} = \sum_i \left(\frac{\partial \mathbf{x}_i}{\partial \mathbf{x}}\right)^T \delta \mathbf{f}_i|_{\delta \mathbf{x}_i} \quad (2)$$

Equation (2) is obtained by taking the directional derivative of (1), assuming that the Jacobian $\partial \mathbf{x}_i / \partial \mathbf{x}$ is constant (i.e. the binding constraint is linear), which is always the case for the barycentric embeddings used here. Nonlinear binding constraints would result in additional terms in equation (2).

From equations (1) and (2) we can write the net force and force differential on a single parent particle as

$$f_k = \sum_{t_i} \sum_{x_j \in t_i} w_k^j f_j^i \quad \text{and} \quad \delta f_k|_{\delta \mathbf{x}} = \sum_{t_i} \sum_{x_j \in t_i} w_k^j \, \delta f_j^i \Big|_{\delta \mathbf{x}_i} \quad (3)$$

where w_k^j is the barycentric weight of particle j with respect to particle k of the parent triangle and f_j^i is the force on particle j from subtriangle t_i. In practice we implement equation (3) by accumulating all forces on both parent and hard bound particles from all their incident elements and subsequently redistributing the force on each hard bound particle to its parents weighted by its barycentric weights. Note that the final force distribution does not depend on the tessellation used but only on the barycentric weights of the hard bound particles.

We use equation (3) for velocity dependent damping forces as well noting that it preserves the symmetry and definiteness of the linear damping forces allowing for our semi-implicit time integration framework. We consider the linear damping model $\bar{\mathbf{f}} = \bar{\mathbf{G}}\bar{\mathbf{v}}$ where $\bar{\mathbf{G}}$ is symmetric negative definite, and the force $\bar{\mathbf{f}}$ and velocity $\bar{\mathbf{v}}$ refer to *all* particles, including hard bound particles. The velocities of this extended set of particles are expressed in terms of the velocities of the parent particles, $\bar{\mathbf{v}} = \mathbf{W}\mathbf{v}$. Using this notation, equation (1) reduces to $\mathbf{f} = \mathbf{W}^T \bar{\mathbf{f}} = \mathbf{W}^T \bar{\mathbf{G}} \mathbf{W} \mathbf{v} = \mathbf{G}\mathbf{v}$ where $\mathbf{G} = \mathbf{W}^T \bar{\mathbf{G}} \mathbf{W}$ preserves both symmetry and negative definiteness of the original damping matrix $\bar{\mathbf{G}}$. A similar derivation shows that the distribution scheme for force differentials given in equation (3) preserves symmetry and negative definiteness of the elastic stiffness matrix, which is an important property for schemes that use an implicit or quasistatic treatment of elastic forces as well.

After forces have been computed and redistributed to the parent particles, accelerations are computed via $\mathbf{a} = \mathbf{M}^{-1}\mathbf{f}$ where $\mathbf{M}$ is diagonal in a typical lumped mass formulation. We store the lumped masses of the parent particles in a vector $\mathbf{m}$, which is the diagonal of $\mathbf{M}$. In analogy to our force derivation, we write the mass vector as the gradient of total momentum with respect to velocity, i.e. $\mathbf{m} = \partial P / \partial \mathbf{v}$. Let $\bar{\mathbf{m}}$ be the masses of all particles. As is typical, one could compute the initial mass of hard bound particles by refining the parent mesh as in figure 2 and assigning one third the mass of each triangle to its vertices, although any scheme for assigning mass (including uniform mass) is allowed. The total momentum is then given as $P = \bar{\mathbf{m}}^T \bar{\mathbf{v}}$, thus $\mathbf{m} = \frac{\partial}{\partial \mathbf{v}} P = \frac{\partial}{\partial \mathbf{v}} \left(\bar{\mathbf{m}}^T \bar{\mathbf{v}}\right) = \frac{\partial}{\partial \mathbf{v}} \left(\bar{\mathbf{m}}^T \mathbf{W} \mathbf{v}\right) = \mathbf{W}^T \bar{\mathbf{m}}$ indicating that masses of hard bound particles should be redistributed to the parents based on the barycentric weights, i.e. the same as for force redistribution. More generally in a non-lumped mass formulation, the mass matrix is given by the Hessian of the kinetic energy as $\mathbf{M} = \mathbf{W}^T \bar{\mathbf{M}} \mathbf{W}$ (see [Sif07]).

Although hard bound particles have no mass or momentum, an *effective mass* is useful for many numerical algorithms. We define the effective mass as the ratio of an applied force to the resultant acceleration of the hard bound particle, i.e. $f_e = m_e a_e$. Denoting the binding weights by w_i, the applied force is distributed to the parent particles via $f_i = w_i f_e$ and the acceleration of the bound particle is $a_e = \sum w_i a_i$. Combining these equations gives

$$\frac{f_e}{m_e} = a_e = \sum w_i a_i = \sum w_i \frac{f_i}{m_i} = f_e \sum \frac{w_i^2}{m_i} \Rightarrow \frac{1}{m_e} = \sum \frac{w_i^2}{m_i} \quad (4)$$

The effective mass is used in the determination of stability restrictions for forces defined on the hard bound particle, e.g. if a hard bound particle is connected to a spring, the effective mass would be used to compute the harmonic mass.

One often needs to apply impulses to hard bound particles, e.g. for collisions. An impulse j_e applied to a hard bound particle is the result of a force f_e acting on the parti-

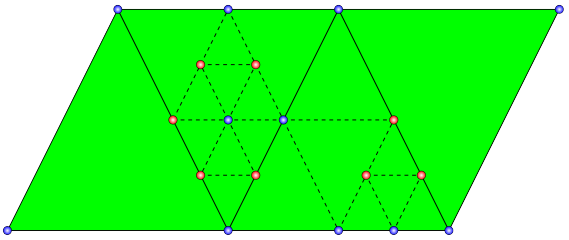

Figure 4: *Two levels of non-graded red refinement leading to T-junctions (red). The T-junctions do not have a complete one-ring of triangles. Our framework treats the T-junction nodes as hard bound particles (red) embedded in the parent mesh (blue).*

cle for an infinitesimal interval Δt, i.e. $j_e = f_e \Delta t$, and from equation (3) we obtain $j_i = w_i j_e$. Furthermore, changes in velocity are governed by $\Delta v_i = j_i/m_i = w_i(m_e/m_i)\Delta v_e$, and displacements follow $\Delta x_i = w_i(m_e/m_i)\Delta x_e$ assuming that $\Delta x_e = \Delta t v_e$.

Figure 3 illustrates the utility of hard bindings in processing collisions. A typical approach to collision processing (due to its simplicity) is to collide the points (possibly only the surface points) of a deformable object with implicitly represented collision geometry, and we collide each tetrahedral mesh node from the sphere's surface with the ground and the kinematically controlled red sphere. Although ground collisions are sufficiently resolved, the kinematically controlled red sphere passes directly through the deformable object, missing any potential collisions with surface particles. Although one might adaptively refine tetrahedra near the colliding red sphere, this increases the total tetrahedra that need to be simulated and the smaller edge lengths lead to a stiffer time step restriction. Our hard binding framework allows us to simply sprinkle particles on the surface of the sphere at a higher density than the tetrahedral mesh for the sake of collisions. This can be done statically or even adaptively based on proximity to collision objects (similar in spirit to [GD04]).

2.1. T-junctions

Figure 4 depicts four coarse triangles undergoing up to two levels of non-graded red refinement leading to a number of T-junctions. Structured adaptive meshing schemes such as [MBTF03] resolve these T-junctions via a combination of red and green refinement. This produces both more elements increasing computational cost as well as lower quality green elements which adversely affect the time step restriction. Additional savings may be realized by exploiting the regularity of adaptive red-only refinements where all elements in the mesh are identical up to rotation and scaling requiring only the storage of a scale factor per element to encode the rest state. For every T-junction, we compute its parents by recursively tracking the endpoints of refined segments. This leads to the barycentric embedding of hard bound particles

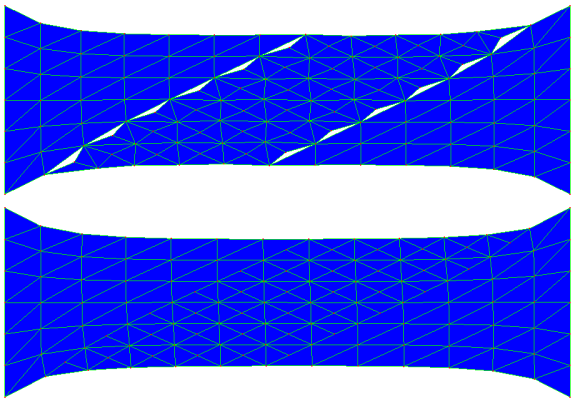

Figure 5: *An elastic sheet meshed with T-junctions is quasistatically stretched without bindings (top) and using hard bindings (bottom).*

on segments or triangles in two spatial dimensions, and on segments, triangles or tetrahedra in three spatial dimensions. Figure 5 illustrates that the straightforward simulation of a T-junction mesh leads to gaps in the mesh (top), while treating T-junction particles as hard bound particles properly constrains them to their incident edges (bottom). We note that T-junctions have also been treated using hierarchical bases (see e.g. [GKS02]).

2.2. Arbitrary embeddings

Although our framework is described in the context of a parent simulation mesh outfitted with a number of embedded particles that will be extended into a fully particle-centric simulation framework (in section 3), here we give an example to illustrate that our framework is more general than this. That is, we switch from the notion of a parent simulation mesh with an embedded set of particles to a parent point cloud simulation system with an embedded mesh. In the context of free-form deformations, one might use a nearest neighbor approach to compute a mesh-free discretization of internal forces on the point cloud, and subsequently move the embedded mesh kinematically. Of course, this can lead to potentially severe distortions of the embedded mesh, especially since the point cloud sampling of deformation is very different from that perceived by the embedded mesh. This can be alleviated by computing the internal forces on the embedded mesh itself, as we do in the elastic ribbon shown in figure 6, and then mapping the resultant forces to the point cloud. Using these forces, we still time integrate the point cloud particles and kinematically enslave the embedded mesh, but the deformation of space is now sampled more adequately for the purpose of simulating the embedded mesh. Note that the kinematic motion of the ribbon is dictated by the k closest parent particles of the point cloud.

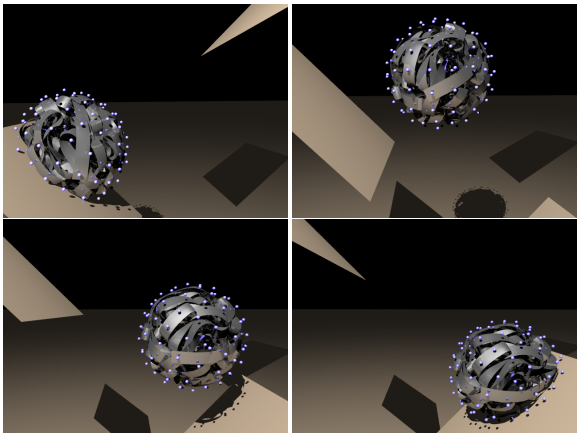

Figure 6: *A ribbon is kinematically embedded in a point cloud. Instead of using a standard mesh-free discretization of the point cloud, we endow the ribbon with a constitutive model which is used to compute elastic forces that are subsequently mapped to the point cloud. The first approach penalizes point cloud deformation and has no direct knowledge of deformation artifacts within the ribbon, while our approach directly penalizes distortions in the ribbon (2.3 sec/frame).*

3. Soft Bindings

The hard bound particle framework is limited because the hard bound particles are kinematically constrained to the parent mesh rather than possessing real degrees of freedom. Thus, specialized algorithms would be required for collision processing, etc. We overcome these issues by introducing the notion of a *soft binding* which couples a new, fully simulation capable particle with an existing parent particle or hard bound particle target location as illustrated in figure 1. We can then ignore hard bound particles for the purpose of collisions and apply collision processing uniformly to parent particles and soft bound particles.

Under no external influence such as collisions or soft bound particle intrinsic forces, soft bound particles should follow their target positions. This is achieved by applying the force that leads to a matching acceleration between the soft bound particle and its target, i.e. the soft bound particles *inherit* elasticity and similar forces from the parent mesh. If the acceleration of a hard bound particle is $a_e = \sum w_i a_i = \sum w_i (f_i/m_i)$, the force that is applied to the soft bound particle to match this acceleration is $f_s = m_s \sum w_i (f_i/m_i)$ where m_s is the mass of the soft bound particle. Note that we only use this process to remap elastic forces, since applying it to damping forces would compromise the symmetry of the damping matrix in our implicit time integration scheme.

The mass of each soft bound particle is set to the *effective mass* of its target (which is just the mass for parent particles). This duplication of mass does not change the effective total mass of the object as measured by applied forces, because our mapping of forces from the parent mesh to the soft bound particles properly accelerates those particles. However, a mass assignment stemming from a consistent physical principle would be desirable.

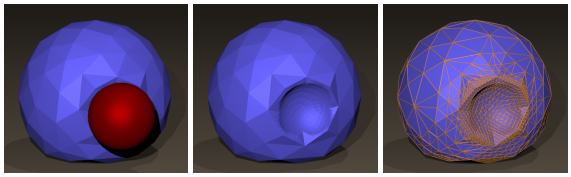

Figure 7: *(Left) A coarse deformable ball collides with a kinematic sphere. (Middle) Soft bindings enable subtetrahedron elasticity in response to the collision. (Right) The dynamically refined surface mesh.*

The coupling strategy between soft bound particles and their targets depends on whether short term or long term drift is desired. Therefore we give separate algorithms for short term and long term drift below.

3.1. Synchronized coupling

The most straightforward way to transfer information from the target particle to the soft bound particle is by directly copying the target state. At first glance, this appears to nullify the desired properties of soft bound particles effectively reducing them to hard bound particles. However, in contrast with hard bound particles which are kept consistent with their target state at all times, soft bound particles are synchronized to their target state at specific points of the time integration loop allowing drift between two subsequent synchronizations. Apart from collision processing, additional forces may also be used to cause drift in soft bound particles, e.g. a soft bound copy of the surface of a volumetric object could be endowed with shell elastic forces. We subsequently propagate any drift from their target locations back to their parents, prior to the next synchronization of soft bindings with their target state.

To propagate drift from a soft bound particle to its target, we compute the discrepancy in position $\Delta x = x - x_e$ and velocity $\Delta v = v - v_e$, where x_e and v_e are the position and velocity of the (possibly hard bound) target particle. If the target particle is part of the parent mesh these increments are directly applied to its position and velocity, while if it is hard bound we use the formulas in section 2 to propagate these increments to the hard bound particle's parent particles. This method for projecting soft bound particles onto their target locations could also be used as a more physically based post-simulation correction to the T-junction nodes in [DMG05].

3.2. Coupling through binding springs

The synchronized coupling scheme only allows for short term drift of the soft bound particles limiting their added simulation capabilities. If long term drift is desired, we instead employ a spring-like force between the soft bound par-

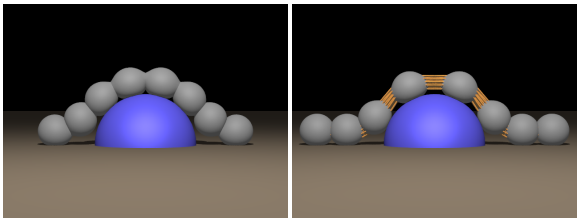

Figure 8: *(Left) Several spheres stitched together using stiff implicit binding springs. (Right) Lowering the binding spring stiffness leads to drift. The binding springs are depicted as orange cylinders (30 sec/frame).*

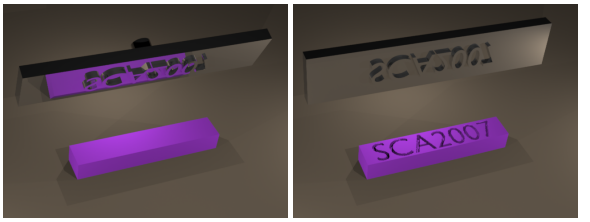

Figure 9: *Allowing the hard bound target positions to drift, one can model subtetrahedron plasticity resulting from collisions of soft bound particles (3 sec/frame).*

ticle and its respective target particle (similar in spirit to a PD controller). This *binding spring* force is

$$f^b = -f_e^b = -k(x - x_e) - b(v - v_e). \tag{5}$$

Note that the binding spring has zero rest length and thus linearizing about the rest state results in a multidirectional (as opposed to the typical directional) damping term. We will capitalize on this fact in our time integration scheme. Binding springs more readily allow for full simulation capabilities in the soft bound particles. One no longer needs to propagate information from the soft bound particles to the parents or to synchronize the soft bound particles, but rather the soft bound particles and parents are now implicitly coupled in a two-way fashion via these binding springs. We emphasize that the important properties of soft bindings depend on the combination of binding springs with force mapping, and would not be achieved through springs alone. Note that a limitation of our approach is that damping forces are not mapped from parents to soft bound particles (to preserve symmetry). Even in the absence of collisions, this leads to drift that has to be corrected via the binding springs.

In figure 7 a coarse tetrahedralized volume has its surface mesh adaptively and dynamically refined based on proximity to collision geometry, and the soft bound particles present at every vertex of the surface mesh provide for subtetrahedron deformation.

3.3. Time integration

Our time integration scheme is a variant of the semi-implicit modified Newmark integration scheme of [ITF04]. The deformable object is evolved from time t^n to t^{n+1} as follows:

1. $\tilde{\mathbf{v}}^{n+\frac{1}{2}} = \mathbf{v}^n + \frac{\Delta t}{2}\mathbf{a}(t^{n+\frac{1}{2}}, \mathbf{x}^n, \tilde{\mathbf{v}}^{n+\frac{1}{2}})$
2. $\tilde{\mathbf{x}}^{n+1} = \mathbf{x}^n + \Delta t\tilde{\mathbf{v}}^{n+\frac{1}{2}}$
3. Process collisions $(\tilde{\mathbf{x}}^{n+1}, \mathbf{v}_n) \rightarrow (\mathbf{x}^{n+1}, \tilde{\mathbf{v}}_n)$
4. * $\mathbf{v}^{n+1} = \tilde{\mathbf{v}}^n + \frac{\Delta t}{2}\left(\mathbf{a}(t^{n+\frac{1}{2}}, \mathbf{x}^{n+\frac{1}{2}}, \tilde{\mathbf{v}}^n) + \mathbf{a}(t^{n+\frac{1}{2}}, \mathbf{x}^{n+\frac{1}{2}}, \mathbf{v}^{n+1})\right)$

where $\mathbf{x}^{n+1/2} = (\mathbf{x}^n + \mathbf{x}^{n+1})/2$. Our scheme deviates from that of [ITF04] in that the trapezoidal velocity update in step 4^* uses $\mathbf{x}^{n+1/2}$. This substitution retains second order accuracy and allows for fully implicit integration of our binding spring forces. Under the common assumption that the velocity dependent forces are linear, we can rewrite step 4^* as

4. $\mathbf{v}^{n+\frac{1}{2}} = \tilde{\mathbf{v}}^n + \frac{\Delta t}{2}\mathbf{a}\left(t^{n+\frac{1}{2}}, \mathbf{x}^{n+\frac{1}{2}}, \mathbf{v}^{n+\frac{1}{2}}\right)$
5. $\mathbf{v}^{n+1} = 2\mathbf{v}^{n+\frac{1}{2}} - \tilde{\mathbf{v}}^n$

The version of trapezoidal rule in step 4^* can suffer from poor numerical conditioning when the two acceleration terms are rather large but of opposite sign, whereas steps 4 and 5 use a completely robust backward Euler update followed by extrapolation. That is, in the limit of a large time step the first acceleration term in step 4^* pushes a positive coefficient c of an eigenvector toward $-\infty$ while the second acceleration term brings that value back from $-\infty$ to $-c$, possibly failing due to loss of precision. However, in the same large time step limit, step 4 damps c to 0 and step 5 robustly extrapolates c through 0 to $-c$.

The time step restriction imposed by a single binding spring is $\Delta t < (b + \sqrt{b^2 + 4\mu k})/k$ where μ is the reduced mass. As is typical, this drives the time step to zero as the spring constant is increased, and thus we propose a fully implicit treatment of the binding springs leveraging the fact that our binding springs are fully linear in both position and velocity (which is not the case for nonzero rest length springs).

Equation (5) can be written in matrix form as $\mathbf{f}^b = -\mathbf{K}\mathbf{x} - \mathbf{B}\mathbf{v}$, where the stiffness and damping matrices $\mathbf{K}$ and $\mathbf{B}$ are symmetric positive semi-definite. We apply our time integration scheme to this force, with the modification that in step 1 we use the half time step position $\tilde{\mathbf{x}}^{n+1/2}$, making this step fully implicit:

$$\begin{aligned}\tilde{\mathbf{v}}^{n+\frac{1}{2}} &= \mathbf{v}^n + \frac{\Delta t}{2}\mathbf{a}^b(t^{n+\frac{1}{2}}, \tilde{\mathbf{x}}^{n+\frac{1}{2}}, \tilde{\mathbf{v}}^{n+\frac{1}{2}}) &(6)\\ &= \mathbf{v}^n + \frac{\Delta t}{2}\left(-\mathbf{M}^{-1}\mathbf{K}\tilde{\mathbf{x}}^{n+\frac{1}{2}} - \mathbf{M}^{-1}\mathbf{B}\tilde{\mathbf{v}}^{n+\frac{1}{2}}\right). &(7)\end{aligned}$$

Substituting $\tilde{\mathbf{x}}^{n+1/2} = \frac{1}{2}(\mathbf{x}^n + \tilde{\mathbf{x}}^{n+1}) = \mathbf{x}^n + \frac{\Delta t}{2}\tilde{\mathbf{v}}^{n+1/2}$, where the last equality comes from step 2, gives

$$\left(\mathbf{I} + \frac{\Delta t}{2}\mathbf{M}^{-1}\mathbf{B} + \frac{\Delta t^2}{4}\mathbf{M}^{-1}\mathbf{K}\right)\tilde{\mathbf{v}}^{n+\frac{1}{2}} = \mathbf{v}^n - \frac{\Delta t}{2}\mathbf{M}^{-1}\mathbf{K}\mathbf{x}^n \tag{8}$$

which is a true backward Euler step implicit in both position and velocity with no time step restriction. We emphasize that equation (6) only replaces step 1 in our time integration loop,

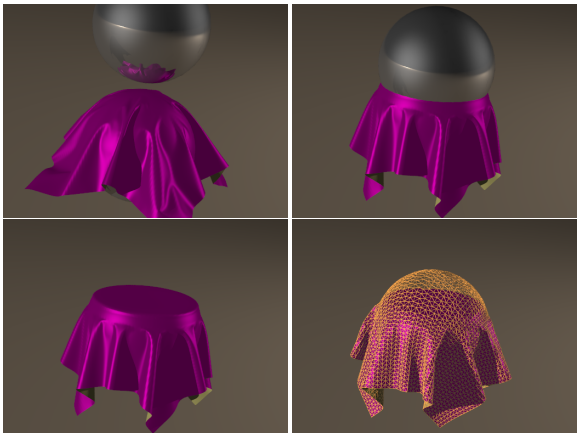

Figure 10: *Cloth pinched between two spheres. Conflicting constraints are resolved by allowing drift of duplicate surfaces using soft bindings (2.8 min/frame, 150×150 mesh).*

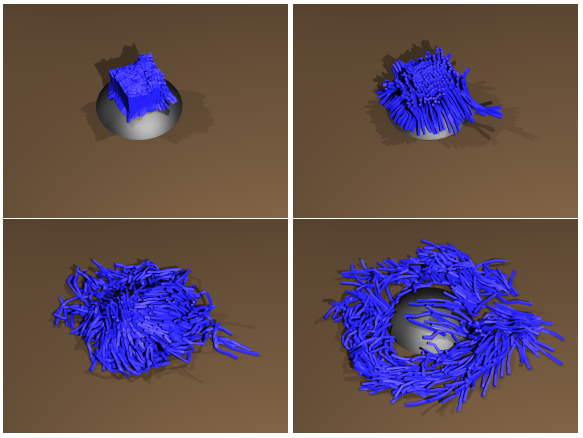

Figure 11: *A coarse* 320 *tetrahedra cube mesh is cut into 289 sticks. Object collisions and self-collisions are resolved by the soft bound embedded surface.*

and step 4 still requires the use of $\mathbf{x}^{n+1/2}$. Our modifications to [ITF04] are crucial for unconditional stability, e.g. using $\mathbf{x}^{n+1}$ instead of $\mathbf{x}^{n+1/2}$ in step 4 leads to a time step restriction of $\Delta t < (b+\sqrt{b^2+8\mu k})/2k$. We stress that this implicit treatment of elastic forces is only used on the binding springs and that all other elastic forces are treated explicitly in order to more accurately resolve dynamic motion.

4. Examples

We used critical damping for the soft binding springs in our examples. The stiffness parameters were set experimentally. Figure 8 shows eight deformable spheres stitched together using implicit binding springs as described in section 3.2. On the left the binding springs are made sufficiently stiff to retain coincidence of the connecting points without incurring additional time step restrictions. The binding springs are visible on the right where their stiffness is reduced. In figure 9 we begin with a coarse deformable block. We then create a duplicate copy of the top surface, which is refined and hard bound to the block. Next, we use binding springs to attach a soft bound duplicate of this refined surface for collision with a stamp. When collisions stretch these binding springs to a length beyond a prescribed threshold the hard bound target particles are moved in material space (and a new barycentric embedding is computed) to bring the length back down to this threshold, analogous to typical plastic yield. We are careful not to move any hard bound target particle outside the material.

Using a variant of the virtual node algorithm [MBF04], we cut a coarse cube mesh consisting of 320 tetrahedra into 289 sticks as shown in figure 11. Each stick is composed of a number of tetrahedra that are only partially filled with material and the polygonization of the material surface is augmented with soft bindings to resolve self-collisions and object collisions. See [SDF07] which is enabled by our new hybrid simulation framework.

Figure 10 shows cloth pinched between two kinematically controlled spheres producing contradictory collision constraints (see [BWK03]). For each particle in the cloth mesh that is pinched between the two spheres, we create two duplicate soft bound particles which each collide with only one of the two spheres. Collisions with the sphere pull these soft bound particles away from their hard bound target locations on the cloth surface resulting in binding spring forces that pull the cloth mesh in both directions equilibrating in a steady state. In the absence of any other forces, the soft bound particles would slide along the sphere in an attempt to minimize the length of the underlying binding springs. However, we map the mesh-based constitutive model forces of the cloth to the soft bound particles making them behave in a fashion similar to their hard bound counterparts, i.e. resisting stretching. Figure 10 (lower right) is a visualization of the effect of mapping the constitutive model forces from the hard bound target locations to their soft bound counterparts. That is, it is similar to creating a mesh which connects all the drifted soft bound particles to each other and to the parent mesh as depicted schematically by the orange wireframe. In fact, one could envision this as simulating three distinct meshes: the parent mesh given by the cloth itself, the orange wireframe mesh, and a second wireframe mesh which agrees with the bottom half of the orange wireframe but is concave up elsewhere resolving collisions with the top sphere. However, we do not need to construct these duplicate meshes, and instead automatically inherit these properties from the parent simulation mesh.

Figure 12 depicts the dynamic simulation of a muscle-driven facial model for speech articulation from [SNF05]. The surface geometry is embedded in a non-graded adaptive red-only BCC mesh with 135K tetrahedral elements (an 85%

reduction from the mesh size in [SSRMF06]. Hard bindings are used to simulate the resulting T-junctions, while soft bound particles are used to model the high resolution surface and also used for collisions, self-collisions and boundary conditions. Since the simulation mesh does not conform to the geometry of the cranium and jawbone, we use soft bound particles on the high-resolution surface to enforce bone attachments as well. A higher stiffness was used to attach the embedded free skin surface to its hard bound counterpart, and a lower stiffness was used for the connections between soft-bound bone attachments and their hard bound targets.

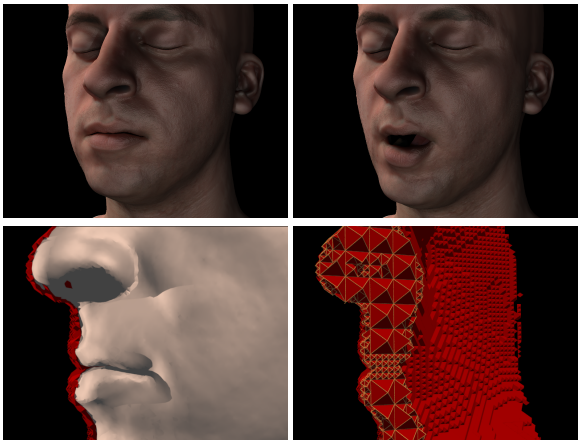

Figure 12: *An adaptive red-only BCC mesh for embedded simulation of a muscle-driven face model. Hard bindings are used for T-junctions, and soft bindings are used for collisions and boundary conditions (18 min/frame).*

5. Extensions to Rigid Body Coupling

Various authors have considered the two-way coupling of rigid and deformable bodies using a variety of schemes, see e.g. [BW97, OZH00, JV03, LF04]. We present preliminary results showing that our hybrid framework can be applied to these types of problems as well.

We consider the rigid body frame (x_c, q), twist (v_c, ω) and inertia tensor $I(q) = R(q) I_0 R(q)^T$, where I_0 is a diagonal inertia tensor and $R(q)$ is the rotation matrix associated with the current orientation. To facilitate our hybrid formulation, we define a *rigid body particle* with the same state as a rigid body. A rigid body hard binding is defined by embedding a particle onto a fixed object space location r_0. Consequently, the kinematic state of the bound particle is given as $x_e = x_c + r(q)$ and $v_e = v_c + \omega \times r(q)$ where $r(q) = R(q) r_0$ is the world space displacement of the embedded particle from the center of mass of the rigid body. We can write $v_e = W(q)(v_c, \omega)$ with the aid of the interpolation matrix $W(q) = (\mathbb{I} \;\; r(q)^{*T})$ where $\mathbb{I}$ is the 3×3 identity matrix and $r(q)^*$ is the cross product matrix. Although x_e cannot be written in similar form, a linearized

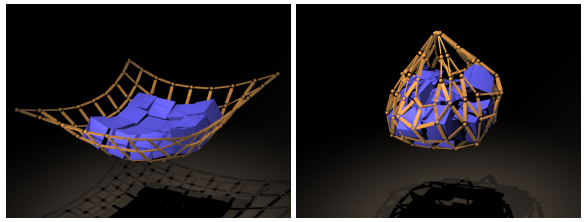

Figure 13: *A net is constructed using binding springs to create simple point joints between hard bound particles on different rigid bodies.*

change in the frame of the rigid body particle admits a similar mapping, i.e. $\Delta x_e = W(q)(\Delta x_c, \Delta q)$, where Δx_c is the displacement of the center of mass and Δq is the vector whose cross product matrix is the linearized rotation satisfying $R(q(t + \Delta t)) = R(q(t)) + \Delta q^* + O(\Delta t^2)$.

Applying a force f_e to the hard bound particle results in a wrench $(f_c, \tau) = W(q)^T f_e$ where $f_c = f_e$ is applied to the center of mass and $\tau = r(q) \times f_e$ is the torque. Next consider a velocity-dependent damping force $f_e = G_e v_e$ on a hard bound particle, which results in $(f_c, \tau) = W(q)^T G_e W(q)(v_c, \omega) = G(v_c, \omega)$ where $G = W(q)^T G_e W(q)$ maps twists to wrenches directly and retains the symmetry and definiteness of the damping matrix G_e. Similarly elastic wrench differentials are given by $(\Delta f_c, \Delta \tau) = W(q)^T K_e W(q)(\Delta x_c, \Delta q) = K(\Delta x_c, \Delta q)$ where $K = W(q)^T K_e W(q)$ is the stiffness matrix expressed in terms of the rigid body particle and $K_e = \partial f_e / \partial x_e$ is the stiffness matrix expressed in terms of the hard bound particle. We use these results to treat rigid body particles in the same fashion as other parent particles in our time integration scheme. Note that the definition of the global damping matrix $\mathbf{G} = \mathbf{W}^T \bar{\mathbf{G}} \mathbf{W}$ and global stiffness matrix $\mathbf{K} = \mathbf{W}^T \bar{\mathbf{K}} \mathbf{W}$ trivially extends to the case of rigid body bindings, by incorporating the interpolation matrix $W(q)$ into the global matrix $\mathbf{W}$ used in these equations. Finally, the global mass matrix $\mathbf{M}$ is augmented with the diagonal entry $\mathrm{diag}(m\mathbb{I}, I)$ for each rigid body particle.

We preliminarily couple our hybrid simulation of rigid body particles to a state-of-the-art rigid body simulation scheme as follows:

1. Copy the state of the rigid bodies to the rigid body particles
2. Compute $\tilde{\mathbf{v}}^{n+\frac{1}{2}}$ as in section 3.3
3. Compute $\tilde{\mathbf{x}}^{n+1}$ using $\tilde{\mathbf{x}}_\mathbf{c}^{n+1} = \mathbf{x}_\mathbf{c}^n + \Delta t \tilde{\mathbf{v}}_\mathbf{c}^{n+\frac{1}{2}}$ and $\tilde{\mathbf{q}}^{n+1} = \mathbf{q}(\Delta t \omega^{n+\frac{1}{2}}) \mathbf{q}^n$
4. Process collisions for deformable objects only
5. Compute $\mathbf{v}^{n+1}$ using steps 4 and 5 from section 3.3
6. Copy *momentum only* from the rigid body particles to the rigid bodies
7. Separately, time integrate and collide the rigid bodies

This last step evolves the rigid bodies forward in time using a standard scheme as in [GBF03], and the two-way cou-

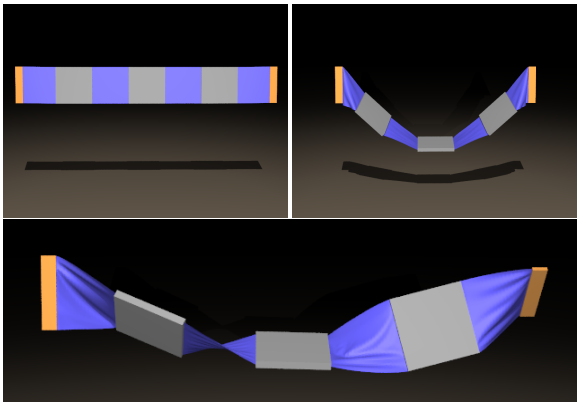

Figure 14: *Coupled simulation of rigid plates and cloth sheets, where cloth particles have been hard bound to the rigid plates. Each of the three rigid plates as well as the two constrained rigid bodies depicted in orange are represented by single rigid body particles.*

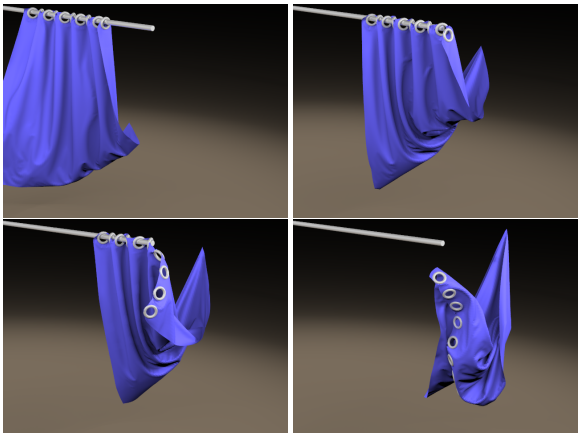

Figure 15: *Simulation of a cloth curtain with a number of particles hard bound on rigid rings that are used to suspend the cloth from a curtain rod. Each of the ten rings is modeled with a single rigid body particle.*

pling is integrated into the standard rigid body solver using steps 1 through 6 where only the effects of momentum are preserved. Although it may be possible to extend the rigid body particles to include more advanced contact, collision, friction, stacking and complex articulation, it is not yet clear how this could be done and we defer it to future work. Figure 13 shows the use of implicitly integrated binding springs to enforce a simple point joint articulation between rigid body hard bound particles. Figures 14 and 15 illustrate two-way coupling between cloth and rigid bodies.

6. Conclusion

We proposed a novel method for augmenting a mesh-based approach to deformable solids simulation with point-based simulation technology. Current limitations include the exclusion of damping terms in the mapping of forces from parents to soft bound particles. Even in the absence of collisions, this leads to drift that has to be corrected via the binding springs. Furthermore, soft bound particles are assigned mass in addition to the mass carried by their parents. Force mapping partially compensates for this by duplicating momentum along with mass on the soft bound particles. Nevertheless, when binding springs are used instead of synchronized coupling, collision handling may be adversely affected by the different amounts of mass assigned to the soft bound surface and its hard bound counterpart. Finally our coupling of deformable and rigid bodies is preliminary and requires separate handling of rigid and deformable body collisions, limiting the applicability of this scheme for scenarios involving complex contact, stacking, friction and articulation.

Acknowledgments

Research supported in part by an ONR YIP award and a PECASE award (ONR N00014-01-1-0620), a Packard Foundation Fellowship, ONR N0014-06-1-0393, ONR N00014-05-1-0479 for a computing cluster, ONR N00014-02-1-0720, ONR N00014-06-1-0505, ARO DAAD19-03-1-0331, NSF CCF-0541148, NSF ITR-0205671, NSF IIS-0326388, NSF ACI-0323866 and NIH U54-GM072970. Special thanks to Rachel Weinstein and Ranjitha Kumar for their help in simulation and rendering.

References

[ACSYD05] ALLIEZ P., COHEN-STEINER D., YVINEC M., DESBRUN M.: Variational tetrahedral meshing. *ACM Trans. Graph. (SIGGRAPH Proc.) 24*, 3 (2005), 617–625.

[BW97] BARAFF D., WITKIN A.: *Partitioned Dynamics*. Tech. rep., Carnegie Mellon University, 1997.

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proc. SIGGRAPH 98* (1998), pp. 43–54.

[BWK03] BARAFF D., WITKIN A., KASS M.: Untangling cloth. *ACM Trans. Graph. (SIGGRAPH Proc.) 22* (2003), 862–870.

[CGC*02a] CAPELL S., GREEN S., CURLESS B., DUCHAMP T., POPOVIĆ Z.: Interactive skeleton-driven dynamic deformations. *ACM Trans. Graph. (SIGGRAPH Proc.) 21* (2002), 586–593.

[CGC*02b] CAPELL S., GREEN S., CURLESS B., DUCHAMP T., POPOVIĆ Z.: A multiresolution framework for dynamic deformations. In *ACM SIGGRAPH Symp. on Comput. Anim.* (2002), ACM Press, pp. 41–48.

[DDCB01] DEBUNNE G., DESBRUN M., CANI M., BARR A.: Dynamic real-time deformations using space & time adaptive sampling. In *Proc. SIGGRAPH 2001* (2001), vol. 20, pp. 31–36.

[DMG05] DEQUIDT J., MARCHAL D., GRISONI L.: Time-critical animation of deformable solids: Collision

detection and deformable objects. *Computer Animation and Virtual Worlds 16*, 3-4 (2005), 177–187.

[GBF03] GUENDELMAN E., BRIDSON R., FEDKIW R.: Nonconvex rigid bodies with stacking. *ACM Trans. Graph. (SIGGRAPH Proc.) 22*, 3 (2003), 871–878.

[GBT06] GISSLER M., BECKER M., TESCHNER M.: Local constraint methods for deformable objects. In *Proc. of the 3rd Workshop in VR Interactions and Physical Simulation (VRIPHYS)* (2006), pp. 1–8.

[GD04] GUY S., DEBUNNE G.: *Monte-Carlo collision detection*. Tech. Rep. RR-5136, INRIA, March 2004.

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: CHARMS: A simple framework for adaptive simulation. *ACM Trans. Graph. (SIGGRAPH Proc.) 21* (2002), 281–290.

[ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible finite elements for robust simulation of large deformation. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* (2004), pp. 131–140.

[JV03] JANSSON J., VERGEEST J.: Combining deformable and rigid body mechanics simulation. *The Vis. Comput. J.* (2003).

[LC06] LOUBÈRE R., CARAMANA E. J.: The force/work differencing of exceptional points in the discrete, compatible formulation of lagrangian hydrodynamics. *J. Comput. Phys. 216* (2006), 1–18.

[LF04] LENOIR J., FONTENEAU S.: Mixing deformable and rigid-body mechanics simulation. In *Comput. Graph. Int.* (june 16-19 2004), pp. 327–334.

[MBF04] MOLINO N., BAO Z., FEDKIW R.: A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. (SIGGRAPH Proc.) 23* (2004), 385–392.

[MBTF03] MOLINO N., BRIDSON R., TERAN J., FEDKIW R.: A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *12th Int. Meshing Roundtable* (2003), pp. 103–114.

[MG04] MÜLLER M., GROSS M.: Interactive virtual materials. In *Graph. Interface* (May 2004), pp. 239–246.

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM Trans. Graph. (SIGGRAPH Proc.)* (2005), 471–478.

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., GROSS M., ALEXA M.: Point based animation of elastic, plastic and melting objects. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* (2004), pp. 141–151.

[MTG04] MÜLLER M., TESCHNER M., GROSS M.: Physically-based simulation of objects represented by surface meshes. In *Proc. Comput. Graph. Int.* (June 2004), pp. 156–165.

[OH99] O'BRIEN J., HODGINS J.: Graphical modeling and animation of brittle fracture. In *Proc. of SIGGRAPH 1999* (1999), pp. 137–146.

[OZH00] O'BRIEN J. F., ZORDAN V. B., HODGINS J. K.: Combining active and passive simulations for secondary motion. *IEEE Comput. Graph. Appl. 20* (2000).

[PKA*05] PAULY M., KEISER R., ADAMS B., DUTRÉ P., GROSS M., GUIBAS L.: Meshless animation of fracturing solids. *ACM Trans. Graph. (SIGGRAPH Proc.) 24*, 3 (2005), 957–964.

[SDF07] SIFAKIS E., DER K., FEDKIW R.: Arbitrary cutting of deformable tetrahedralized objects. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim. (in press)* (2007).

[SHGS06] STEINEMANN D., HARDERS M., GROSS M., SZEKELY G.: Hybrid cutting of deformable solids. In *Proc. of the IEEE Virtual Reality Conference* (2006), pp. 35–42.

[Sif07] SIFAKIS E.: *Algorithmic aspects of the simulation and control of computer generated human anatomy models*. PhD thesis, Stanford University, June 2007.

[SNF05] SIFAKIS E., NEVEROV I., FEDKIW R.: Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph. (SIGGRAPH Proc.)* (2005).

[SOG06] STEINEMANN D., OTADUY M. A., GROSS M.: Fast arbitrary splitting of deforming objects. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* (2006), pp. 63–72.

[SSRMF06] SIFAKIS E., SELLE A., ROBINSON-MOSHER A., FEDKIW R.: Simulating speech with a physics-based facial muscle model. *ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* (2006), 261–270.

[TF88a] TERZOPOULOS D., FLEISCHER K.: Deformable models. *The Vis. Comput. 4*, 6 (1988), 306–331.

[TF88b] TERZOPOULOS D., FLEISCHER K.: Modeling inelastic deformation: viscoelasticity, plasticity, fracture. *Comput. Graph. (SIGGRAPH Proc.)* (1988), 269–278.

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. *Comput. Graph. (Proc. SIGGRAPH 87) 21*, 4 (1987), 205–214.

[TSSB*05] TERAN J., SIFAKIS E., SALINAS-BLEMKER S., NG-THOW-HING V., LAU C., FEDKIW R.: Creating and simulating skeletal muscle from the visible human data set. *IEEE Trans. on Vis. and Comput. Graph. 11*, 3 (2005), 317–328.

[WSG05] WICKE M., STEINEMANN D., GROSS M.: Efficient animation of point-sampled thin shells. In *Proc. of Eurographics* (2005), vol. 24.