Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2008)
M. Gross and D. James (Editors)

Two-way Coupling of Rigid and Deformable Bodies

Tamar Shinar[†] Stanford University

Craig Schroeder[‡] Stanford University, Pixar Animation Studios

Ronald Fedkiw[‡] Stanford University, Industrial Light + Magic

Abstract

We propose a framework for the full two-way coupling of rigid and deformable bodies, which is achieved with both a unified time integration scheme as well as individual two-way coupled algorithms at each point of that scheme. As our algorithm is two-way coupled in every fashion, we do not require ad hoc methods for dealing with stability issues or interleaving parts of the simulation. We maintain the ability to treat the key desirable aspects of rigid bodies (e.g. contact, collision, stacking, and friction) and deformable bodies (e.g. arbitrary constitutive models, thin shells, and self-collisions). In addition, our simulation framework supports more advanced features such as proportional derivative controlled articulation between rigid bodies. This not only allows for the robust simulation of a number of new phenomena, but also directly lends itself to the design of deformable creatures with proportional derivative controlled articulated rigid skeletons that interact in a life-like way with their environment.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling, Physically based modeling

Keywords: rigid bodies, deformable solids, two-way coupling

1. Introduction

A number of authors have proposed methods for simulating two-way coupling between rigid and deformable bodies, see e.g. [BW97, OZH00, JV03, LF04, SSIF07b]. Although coupling deformable and rigid bodies is interesting from the standpoint of simulating new phenomena as demonstrated by those authors, it is also quite interesting from the standpoint of designing creatures. Typically creatures are designed as articulated rigid bodies with some sort of joint or muscle control with notable examples being Luxo Jr. [WK88], athletes from the 1996 Summer Olympics [HWBO95], and the virtual stunt man of [FvT01]. However, creatures are more life-like when their internal skeleton can be used to drive a deformable exterior, such as the tentacles for Davy Jones [CDH06, Wei07]. The difficulty with wrapping a rigid skeleton with a deformable exterior is that the creature can only interact with its environment if environmental forces deform the exterior, and the deformable exterior subsequently applies forces to the rigid interior. That is, modeling deformable creatures with rigid skeletons requires two-way rigid/deformable interaction [GOT*07].

For the two-way coupling of physical phenomena, there are two common approaches. One approach is to start with the best methods for each phenomenon and subsequently design methods for linking these disparate simulation techniques together. Typically interleaving is necessary, where each simulation is run using the results of the previous one in an alternating one-way coupled fashion. This leads to stability issues as one system cannot adequately predict what the other system will do–similar to explicit time integration. Various ad hoc tricks can be applied to increase the stability making the system semi-implicit (e.g. solving for the effect of damping forces between rigid and deformable bodies implicitly, and subsequently mapping the momentum back to the rigid body simulator [SSIF07b]). Still, stability issues remain. The other approach is to design a fully two-way coupled system, and there are essentially two ways to accomplish this. One could use the same type of simulation for both types of phenomena (e.g. SPH for both solids and liquids [MST*04] or cloth as an articulated net of rigid bodies [Ben07]). Unfortunately, this typically leads to inferior methods for at least one of the two phenomena being simulated. For example, one could imagine extending the mass-spring simulation framework to the rigid limit and simulating rigid bodies with very stiff springs. We take the alternative approach and fully hybridize the simulation frameworks in a manner that maintains the ability to use the more advanced techniques for each physical phenomenon. That is, we want to retain the ability of our rigid body framework to accurately model contact, collision, stacking, friction, articulation, proportional derivative (PD) control, etc., and our deformable object framework to handle arbitrary constitutive models, finite elements, masses and springs, volumes or thin shells, contact, collision, self-collision, etc. In addition, we would like to maintain stability and avoid ad hoc methods for interleaving the simulations.

Full two-way coupling of state-of-the-art rigid body and deformable object time integration schemes, though seem-

[†] email: shinar@stanford.edu
[‡] email: {cas43|fedkiw}@cs.stanford.edu

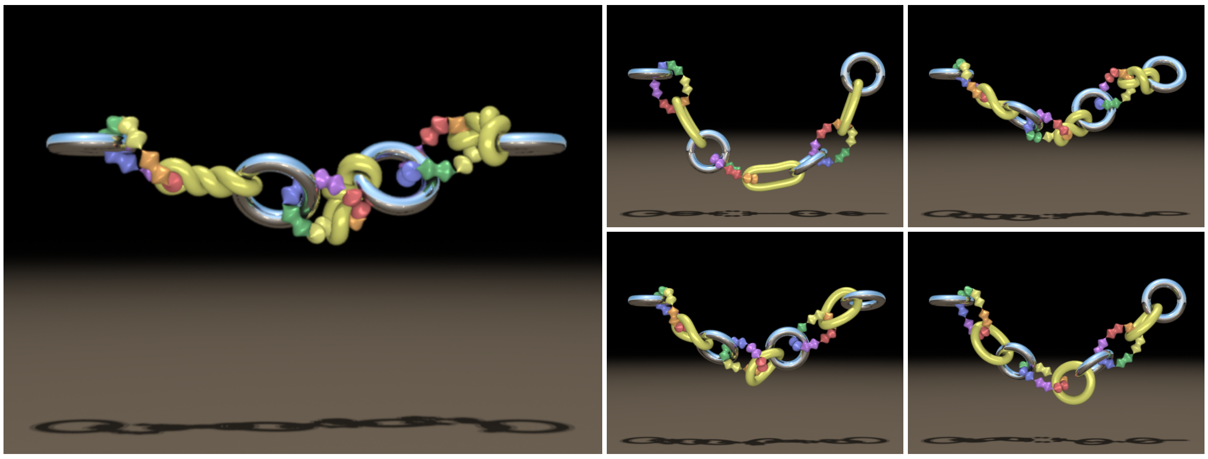

Figure 1: *Chain composed of both rigid and deformable tori, as well as loops of articulated rigid bodies. The leftmost torus is static, and the rightmost torus is kinematically spun to wind and unwind the chain.*

ingly straightforward, is more difficult and subtle than it appears. Robust rigid body systems tend to be sensitive to the time integration scheme, because a minor change in the scheme can, for example, cause blocks to tumble rather than slide down an inclined plane. In contrast, deformable object Newmark schemes are far less sensitive and allow more freedom in the way collisions are processed but utilize separate position and velocity updates as well as implicit solves. Our approach demonstrates that we can fully integrate these disparate methods without losing their individual capabilities. Notably, this can be done by coupling together standard rigid body and standard deformable body simulation systems using our newly proposed techniques instead of requiring one to design a unique simulation system from scratch.

We propose a unified time integration scheme, where the rigid and deformable bodies are integrated forward together in every manner, not only with position and velocity evolution, but such that collision, contact, interpenetration resolution, etc. are all handled at the fine-grained level with fully two-way coupled algorithms. On the rigid body side, we were guided by the time integration scheme proposed by [GBF03], which proposes a clean separation of contact and collision and handles both simple and more complex scenarios ranging from a block sliding down an inclined plane to large-scale stacking behaviors. We also wanted our rigid body simulator to include the effects of articulation and internally torque controlled joints [WTF06, WGF08]. On the deformable side, we required a Newmark-style algorithm as in [SSIF07b], which cleanly separates the evolution of position and velocity. The deformable algorithm should at least be implicit on the damping forces as in [BFA02] but could also be fully implicit as in [BW98]. We propose such a time integration scheme, which evolves every object synchronously using fully coupled algorithms at each step.

We describe in detail the steps of our unified time integration scheme below. We note, however, that the approach does not depend on our particular choice of algorithms for the various subsystems but rather serves as a proof of concept of the benefits of a unified approach. Other choices of specific algorithms, e.g., for contact and collision, can be substituted. Much of what we incorporate is optional, including the incompressibility of [ISF07], the articulation of [WTF06], the PD control of [WGF08], and the bindings of [SSIF07b]. We include them to demonstrate the breadth of phenomena that can be incorporated.

2. Time Integration

The basic structure of our time integration scheme is that of a Newmark method. Newmark methods are characterized by separate position and velocity updates. In particular, the velocity used in the position update can be distinct from the final velocity, and we take advantage of this to treat the position and velocity updates differently. For example, one might add constraint violating components to the velocity during the position update to correct drift, while projecting out these components in the final velocity update (e.g. as in [ISF07]). Moreover, maintaining the Newmark structure allows us to incorporate a wide range of algorithms from computational mechanics. For example, we can enforce the incompressibility of materials, deal with contact and collisions, self-collisions, friction, etc. For background on the Newmark family of methods see e.g. [Hug00].

We begin by outlining our time integration scheme as composed of five major steps:

I. Advance velocity $\mathbf{v}^n \longrightarrow \tilde{\mathbf{v}}^{n+\frac{1}{2}}$
II. Apply collisions $\mathbf{v}^n \longrightarrow \hat{\mathbf{v}}^n$, $\tilde{\mathbf{v}}^{n+\frac{1}{2}} \longrightarrow \hat{\mathbf{v}}^{n+\frac{1}{2}}$
III. Apply contact and constraint forces $\hat{\mathbf{v}}^{n+\frac{1}{2}} \longrightarrow \mathbf{v}^{n+\frac{1}{2}}$
IV. Advance positions $\mathbf{x}^n \longrightarrow \mathbf{x}^{n+1}$ using $\mathbf{v}^{n+\frac{1}{2}}$, $\hat{\mathbf{v}}^n \longrightarrow \bar{\mathbf{v}}^n$
V. Advance velocity $\bar{\mathbf{v}}^n \longrightarrow \mathbf{v}^{n+1}$

Adhering to the time integration scheme proposed in [GBF03], we cleanly separate position and velocity updates

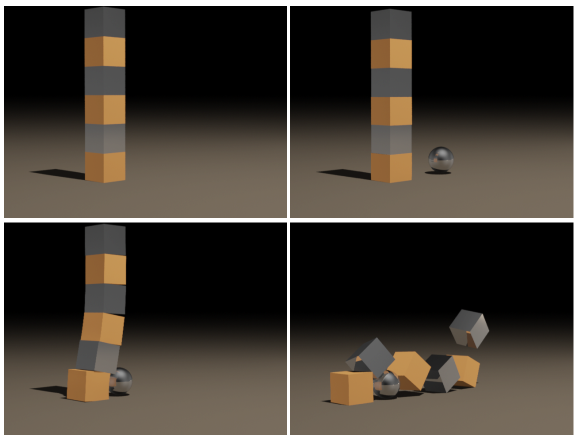

Figure 2: *Silver rigid boxes and orange deformable boxes are arranged to form a stack, which is knocked down by a rigid ball.*

from contact and collision. Collisions are done with time t^n velocities so that objects in contact do not bounce, and contacts are performed only on a subset of pairs processed by collisions so that one does not inaccurately apply contact forces to objects before they collide. The algorithmic ordering of collisions followed by contact followed by position updates followed by velocity updates is exactly as in [GBF03] except for the initial advancement of velocities to $\mathbf{v}^{n+1/2}$. In fact, their algorithm would have been more accurate if it had done this, as opposed to using the full Δt for the predictive velocities, obtaining $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^n + \Delta t^2 \mathbf{a}$. [WGF08] took special measures to deal with this inaccuracy, changing parameters to get the desired solution. Using $\mathbf{v}^{n+1/2}$ as in the Newmark scheme automatically alleviates this issue.

Each of the five major steps of the time integration scheme is further described in the five sections that follow.

3. Step I: Advance Velocity

Typically the first step in a Newmark iteration scheme is to predict the velocity that will be used to update the positions. We accomplish this as follows:

1. Advance velocities $\mathbf{v}_\star^{n+\frac{1}{2}} = \mathbf{v}^n + \frac{\Delta t}{2}\mathbf{a}(t^{n+\frac{1}{2}}, \mathbf{x}^n, \mathbf{v}_\star^{n+\frac{1}{2}})$
2. Apply volume correction $\mathbf{v}_\star^{n+\frac{1}{2}} \rightarrow \mathbf{v}_{\star\star}^{n+\frac{1}{2}}$
3. Apply self-repulsions $\mathbf{v}_{\star\star}^{n+\frac{1}{2}} \rightarrow \tilde{\mathbf{v}}^{n+\frac{1}{2}}$

Here and in the remainder of the paper we use starred variables to denote intermediate states that do not carry over from section to section. When using semi-implicit time integration with implicit, linear damping forces as in [BMF03], step 1 requires the solution of a linear system, which we solve using the conjugate gradient method. One could instead use the fully implicit method from [BW98]. In that case step 1 becomes $\mathbf{v}_\star^{n+1/2} = \mathbf{v}^n + \frac{\Delta t}{2}\mathbf{a}(t^{n+1/2}, \mathbf{x}^{n+1/2}, \mathbf{v}_\star^{n+1/2})$, where $\mathbf{x}^{n+1/2}$ is replaced with

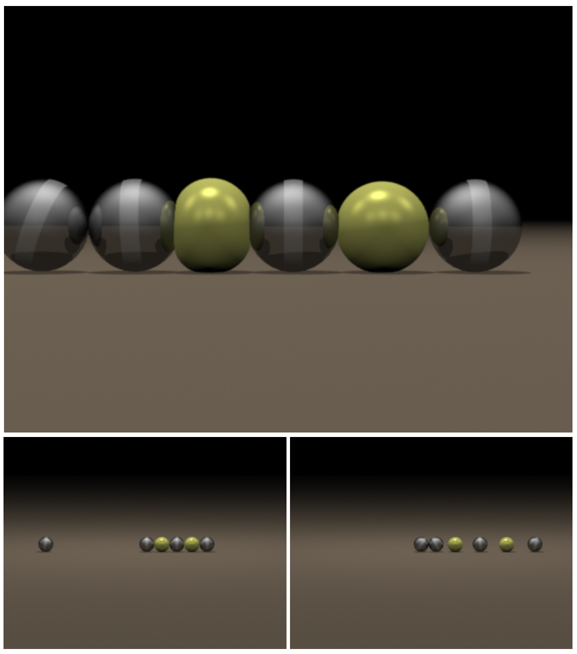

Figure 3: *A row of silver rigid balls and yellow deformable balls are impacted by a rigid ball. The yellow balls are modeled as deformable and incompressible materials.*

$\mathbf{x}^{n+1/2} = \mathbf{x}^n + \frac{\Delta t}{2}\mathbf{v}_\star^{n+1/2}$. The resulting nonlinear equation is subsequently solved via Newton-Raphson iteration. During the solve, the acceleration in step 1 is projected to satisfy equality constraints such as joint velocity equality constraints, but we avoid applying constraints that may lead to undesirable sticking artifacts at this stage.

If we are simulating incompressible solids, as in Figure 3, step 2 adjusts the velocities such that local volume errors will be corrected when positions are advanced [ISF07]. One could also make the velocity field inextensible as in [GHF*07]. Step 3 applies self-repulsions as in [BFA02].

While we include steps 2 and 3 here, the only essential part of Step I is to advance the velocity to time $t^{n+1/2}$ as required by the Newmark structure.

4. Step II: Collisions

Once the time $t^{n+1/2}$ velocities have been computed, we correct them for rigid/rigid, rigid/deformable and deformable/deformable collisions. As in [GBF03], collisions are processed before contact to allow elastically colliding rigid bodies to bounce. The main goal of this step is to adjust velocities for collisions, and other collision algorithms may be used. The steps we use for our collision processing are

1. Process collisions $\mathbf{v}^n \rightarrow \mathbf{v}_\star^n$
2. Apply post-stabilization $\mathbf{v}_\star^n \rightarrow \hat{\mathbf{v}}^n$
3. Re-evolve velocities $\hat{\mathbf{v}}^{n+\frac{1}{2}} = \hat{\mathbf{v}}^n + (\tilde{\mathbf{v}}^{n+\frac{1}{2}} - \mathbf{v}^n)$

For collision detection, all positions are evolved forward in time to look for interferences between pairs of rigid/rigid,

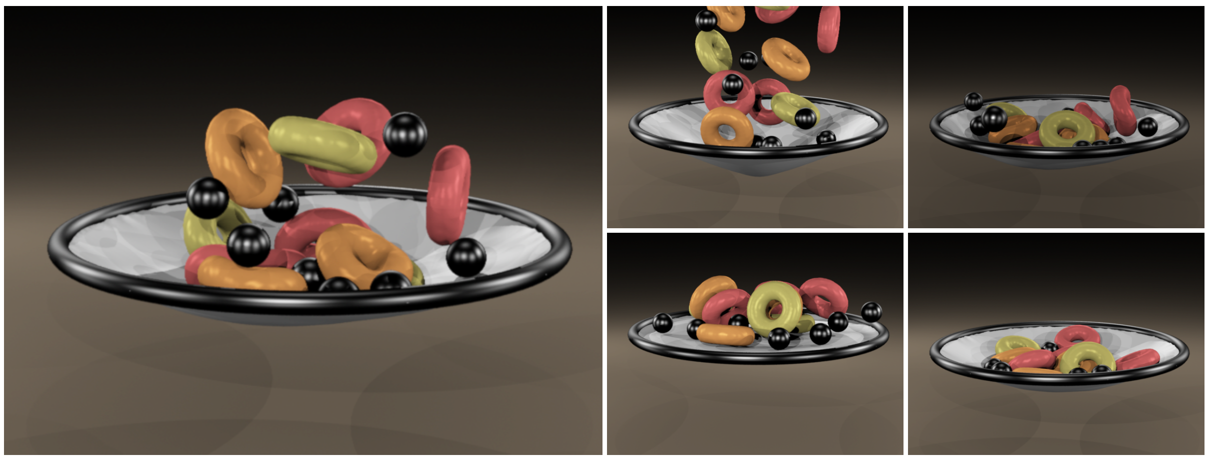

Figure 4: *Ten rigid balls and ten deformable tori fall on a cloth trampoline. The trampoline is constructed by embedding the cloth in a kinematically controlled torus, which tosses the objects after they have landed and then allows them to settle.*

rigid/deformable, or deformable/deformable bodies using the update formula $\mathbf{x}_\star^{n+1} = \mathbf{x}^n + \Delta t \tilde{\mathbf{v}}^{n+\frac{1}{2}}$ (see Section 4.1 for rigid body orientations). As in [GBF03], collisions are processed using $\mathbf{v}^n$ so that objects in contact do not bounce even when they have nonzero coefficient of restitution. For details on rigid/rigid collisions, see [GBF03]. Rigid/deformable collisions are processed by iteratively colliding each particle of the deformable body against the rigid body. Each rigid/particle pair is processed using the rigid/rigid collision algorithm treating the particle as a rigid body (with an infinite inertia tensor) and a zero coefficient of restitution. For deformable/deformable pairs, one could use tetrahedral collisions as in [TSIF05], however we instead use the self-collisions of [BFA02] after the position update as described in Section 6.

In Gauss-Seidel fashion, we iteratively process the collisions one pair at a time and re-update the post-collision position of each body via $\mathbf{x}_\star^{n+1} = \mathbf{x}^n + \Delta t(\mathbf{v}_\star^n + (\tilde{\mathbf{v}}^{n+1/2} - \mathbf{v}^n))$ where $\tilde{\mathbf{v}}^{n+1/2} - \mathbf{v}^n$ includes the forces added in step 1 of Section 3. If no collision was applied, then $\mathbf{v}_\star^n = \mathbf{v}^n$, and the position is unchanged. Note that the positions computed here are only used to resolve collisions and are subsequently discarded. We also explicitly save a list of all pairs processed

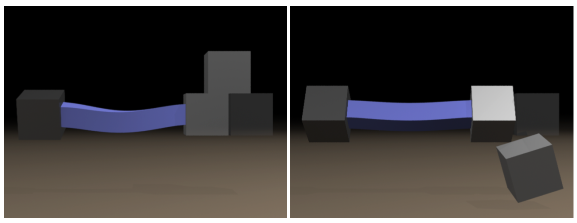

Figure 5: *The four silver boxes are rigid, and the deformable blue bar is attached to the boxes using the embedding framework of [SSIF07b]. The far left box is kinematically rotated, causing the bar to twist and subsequently turn the box to its right, which is attached to the static box to the right of it by a twist joint. The free box on top falls off as the bar is twisted.*

so that contact later considers only those pairs already processed by collisions. This ensures that newly colliding objects will have an opportunity to bounce before being processed for contact. After processing collisions, we apply the post-stabilization algorithm of [WTF06] to ensure that the collision velocities do not violate joint constraints. Finally, step 3 applies the effects of collisions $\hat{\mathbf{v}}^n - \mathbf{v}^n$ to the time $t^{n+1/2}$ velocities $\tilde{\mathbf{v}}^{n+1/2}$.

4.1. Second Order Rigid Body Evolution

Given a rigid body in a certain orientation with a certain kinetic energy and angular momentum, the number of states the rigid body is allowed to evolve to in the absence of external torque is limited by conservation of both energy and momentum. Not every orientation will conserve both quantities, and typical simulators only conserve momentum, allowing kinetic energy to rise or fall. [WTF06] used the first order update $q^{n+1} = \hat{q}(\Delta t\omega)q^n$ where $\hat{q} = (\cos(|\omega|/2), \sin(|\omega|/2)\omega/|\omega|)$ is the unit quaternion which rotates by $|\omega|$ around the vector ω. To reduce errors in kinetic energy, we instead propose using the second order update $q^{n+1} = \hat{q}(\Delta t\omega + (1/2)\Delta t^2 I^{-1}(L\times\omega))q^n$. A straightforward Taylor expansion can be used to verify that this formula is second order accurate, and we note that it essentially uses a time $t^{n+1/2}$ angular velocity ω similar to Newmark methods. The Taylor expansion is simplified by observing that $\hat{q}(\omega)$ is equivalent to e^{ω^*}, where ω^* is the cross product matrix of ω.

5. Step III: Contact and Constraint Forces

The main purpose of this section is contact and contraint processing. Articulation and PD are optional, and other contact processing algorithms could be substituted. The steps used for the contact and constraint forces stage are

1. Compute contact graph using $\hat{\mathbf{v}}^n$ and $\hat{\mathbf{v}}^{n+\frac{1}{2}}$
2. Apply PD to velocities $\hat{\mathbf{v}}^{n+\frac{1}{2}} \to \mathbf{v}_\star^{n+\frac{1}{2}}$

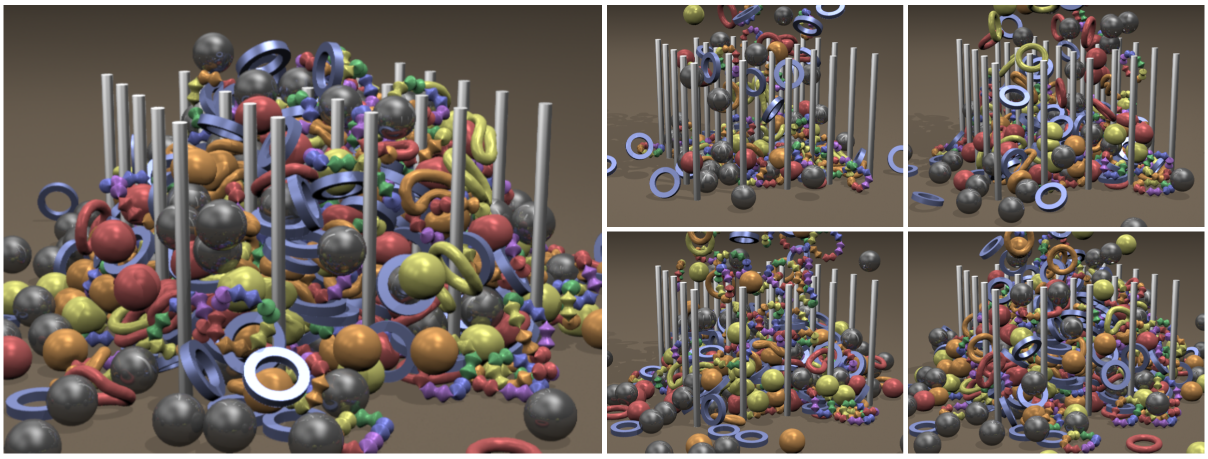

Figure 6: *1600 bodies are dropped on a 5×5 grid of static rigid pegs to form a pile. There are 160 colored deformable balls, 160 colored deformable tori, 160 rainbow-colored articulated loops with six rigid bodies each, 160 silver rigid balls and 160 blue rigid rings.*

3. Apply contact and pre-stabilization $\mathbf{v}_\star^{n+\frac{1}{2}} \to \mathbf{v}^{n+\frac{1}{2}}$

We compute a contact graph for the rigid bodies similar to [GBF03] which also includes the order in which articulated rigid bodies will be processed for pre-stabilization as in [WTF06]. We also apply PD control, see [WGF08]. Proceeding in the order determined by the contact graph, each rigid body is processed in turn for contact with all rigid and deformable bodies it interpenetrates. For rigid/rigid pairs, we perform contact as in [GBF03]. Rigid/deformable pairs are treated in the same manner as they were treated for collisions, since we always treat deformable body collisions inelastically. This process is iterated as in [GBF03]. If one were using tetrahedral collisions for deformable/deformable pairs [TSIF05], they would also be processed at this stage. We instead apply the self-collisions of [BFA02] after the position update as described in Section 6 to resolve deformable/deformable contact. The stack in Figure 2 demonstrates the effectiveness of the two-way contact algorithm, standing upright until being struck at its base by a rigid ball.

5.1. Improved Pre-stabilization

[WTF06] performed pre-stabilization on orientations by solving a quaternion equation of the form $f(\mathbf{j}_\tau) = \mathbf{q}_p(\mathbf{j}_\tau) - \mathbf{q}_c(\mathbf{j}_\tau) = 0$ using Newton-Raphson iteration. To alleviate issues with quaternion subtraction, we instead solve $\mathbf{g}(\mathbf{j}_\tau) = \mathbf{q}_p\mathbf{q}_c^{-1} = (\pm 1, \mathbf{0})$. $\mathbf{g}(\mathbf{j}_\tau)$ consists of both a scalar and vector part, and it turns out to be enough to solve the nonlinear equation that sets the vector part equal to $\mathbf{0}$.

6. Step IV: Advance Positions

In the preceeding steps, we made all the desired adjustments to the velocities. The purpose of Step IV is then to evolve the bodies to their final positions, which we do as follows:

1. Advance positions $\hat{\mathbf{x}}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^{n+\frac{1}{2}}$
2. Interpenetration resolution $\hat{\mathbf{x}}^{n+1} \to \mathbf{x}^{n+1}$
3. Post-stabilization $\hat{\mathbf{v}}^n \to \overline{\mathbf{v}}^n$

Since our rigid/rigid and rigid/deformable contact algorithms do not guarantee that the bodies are completely interpenetration free, we apply an interpenetration resolution method (similar in spirit to [Bar96]), as described in detail below. Finally, since steps 1 and 2 changed rigid body orientations, rigid body angular velocities also changed, and thus post-stabilization of velocities for articulated rigid bodies must be applied.

6.1. Interpenetration Resolution

One might consider an approach to resolving penetration that operates by iterating pairwise algorithms. This approach suffers from the *messenger problem*, where a large amount of momentum must be exchanged through low-capacity messengers. Consider a row of colliding objects, with the outer two objects being massive and the objects between them being light. The collision must be resolved by moving the massive outer bodies, which requires the exchange of a large impulse. Because the objects in between are light, they are unable to transfer large impulses without obtaining high velocities. Each collision processing step is only able to transfer an impulse across the chain one body at a time, so a large number of iterations are required to complete the momentum transfer.

With this in mind, we propose a novel method to remove small interpenetrations between bodies. Rather than process pairwise as was done for the contact and collision algorithms, we consider one central object (either a rigid body or a particle of a deformable body) at a time and push out every outer object (rigid or deformable) intersecting it simultaneously in a fully two-way coupled fashion. This helps address the messenger problem by allowing a single step to transfer impulses two links at a time instead of one and by avoiding the problem entirely when there is only one such light body. This approach is not easily applied to the full contact and

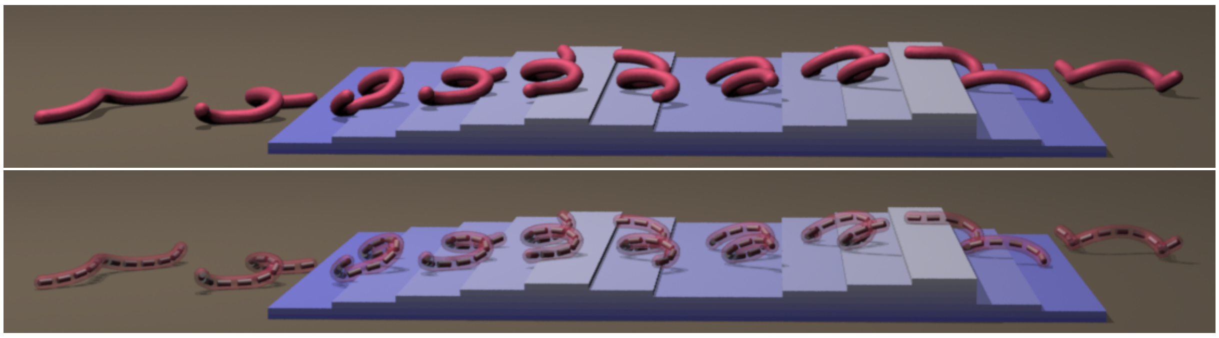

Figure 7: *A snake constructed by embedding a 12-joint articulated rigid body skeleton in a deformable body. The snake sidewinds up and down stairs using PD-control on its skeleton.*

collision problems, primarily because of the complications introduced by friction. The central body c exchanges an impulse $\mathbf{j}_o$ with an outer body o resulting in a change in relative velocity at the pair's intersection point of

$$m_c^{-1}\mathbf{j} + \mathbf{r}_o^{*T} I_c^{-1}\mathbf{j}_\tau - K_o\mathbf{j}_o = \mathbf{v}_o \tag{1}$$

where $K_o = m_o^{-1}\delta + \mathbf{q}_o^{*T} I_o^{-1}\mathbf{q}_o^*$, $\mathbf{r}_o$ and $\mathbf{q}_o$ are the vectors from the center of masses of body c and body o to the point of application of $\mathbf{j}_o$, respectively, and $\mathbf{j} = -\sum_o \mathbf{j}_o$ and $\mathbf{j}_\tau = -\sum_o \mathbf{r}_o^*\mathbf{j}_o$ are the net linear and angular impulses applied to body c. Solving Equation (1) for $\mathbf{j}_o$ and plugging the result into the expressions for $\mathbf{j}$ and $\mathbf{j}_\tau$ gives the symmetric 6×6 system

$$\begin{pmatrix} m_c\delta + \sum\limits_o K_o^{-1} & \sum\limits_o K_o^{-1}\mathbf{r}_o^{*T} \\ \sum\limits_o \mathbf{r}_o^* K_o^{-1} & I_c + \sum\limits_o \mathbf{r}_o^* K_o^{-1}\mathbf{r}_o^{*T} \end{pmatrix} \begin{pmatrix} m_c^{-1}\mathbf{j} \\ I_c^{-1}\mathbf{j}_\tau \end{pmatrix} = \begin{pmatrix} \sum\limits_o K_o^{-1}\mathbf{v}_o \\ \sum\limits_o \mathbf{r}_o^* K_o^{-1}\mathbf{v}_o \end{pmatrix}. \tag{2}$$

Once Equation (2) is solved for the net impulse $\mathbf{j}$ and $\mathbf{j}_\tau$ on the central body, each $\mathbf{j}_o$ is obtained from substitution into Equation (1). To achieve a desired separation distance $\mathbf{d}_o$, we take $\mathbf{v}_o = \mathbf{d}_o/\Delta\tau$ and integrate positions with the velocity change due to the computed impulses for a pseudo-time of $\Delta\tau$. The above procedure is iterated over the bodies and is both more accurate and converges faster than an analogous pairwise method.

Equations (1) and (2) apply generally to both rigid bodies and deformable particles, with the simplification that $\mathbf{r}_o = \mathbf{0}$ for a central deformable particle (reducing Equation (2) to a 3×3 system) and that $\mathbf{q}_o = \mathbf{0}$ for an outer deformable particle. When the central body is static or kinematic (having infinite inertia) the equations for the outer bodies decouple as the first two terms in Equation (1) vanish. When any outer bodies are static or kinematic the system must be decomposed into the degrees of freedom determined by the static bodies and the remaining degrees of freedom corresponding to the nullspace of the equations for the static bodies. Afterwards, the system can be reassembled and solved. For more details see [Shi08].

7. Step V: Advance Velocity

The second half of the Newmark time integration scheme is the velocity update, which we do as follows:

1. Make incompressible $\overline{\mathbf{v}}^n \longrightarrow \mathbf{v}_\star^n$
2. $\mathbf{v}_\star^{n+1} = \mathbf{v}_\star^n + \Delta t\,\hat{\mathbf{a}}(t^{n+\frac{1}{2}}, \frac{1}{2}(\mathbf{x}^n + \mathbf{x}^{n+1}), \frac{1}{2}(\mathbf{v}_\star^n + \mathbf{v}_\star^{n+1}))$
3. $\mathbf{v}_{\star\star}^{n+1} = \mathbf{v}_\star^n + \Delta t\,\mathbf{a}(t^{n+\frac{1}{2}}, \frac{1}{2}(\mathbf{x}^n + \mathbf{x}^{n+1}), \frac{1}{2}(\mathbf{v}_\star^n + \mathbf{v}_\star^{n+1}))$
4. Apply constraints: post-stabilization, PD control, contact, post-stabilization and self-repulsions $\mathbf{v}_{\star\star}^{n+1} \longrightarrow \mathbf{v}^{n+1}$

We project the velocity field for incompressibility before our velocity evolution as in [ISF07]. In step 2, we use a variant of trapezoid rule to solve for the velocities. In practice, we break this step into a backward Euler solve followed by extrapolation as in [SSIF07b]. We note that one could instead use backward Euler, which introduces more damping. We discuss steps 2-4 in more detail below.

7.1. Constrained Solve

The linear system in step 2 is solved using conjugate gradients. We include as many linear constraints as possible in the conjugate gradient solve, accounting for post-stabilization of joints between rigid bodies and two-way contact. This is desirable because it allows the effects of constraints to propagate instantly during the implicit solve. Note that $\hat{\mathbf{a}}$ has been used instead of $\mathbf{a}$ in step 2 above to indicate that these forces are modified during the conjugate gradient solve by applying projections to satisfy constraints as in [BW98].

7.2. Final Velocities

As shown in [BW98], the unprojected residual of the converged system in step 2 gives the net normal force due to all constraints. [BW98, SSIF07a] use the unprojected residual to compute frictional forces due to contact between particles and a single immovable body. They are able to postprocess each particle independently based on the normal force it feels because they enforce contact in a one-way fashion. That is, computing the frictional forces on a particle results in a change in the particle's tangential velocity relative to the body, but does not affect the body's velocity. Thus, the particle/body interactions are decoupled, with the body acting as though it had infinite inertia.

In contrast to [BW98, SSIF07a], the interactions we compute are fully two-way. Thus, we cannot postprocess constrained particles for friction independently after the solve,

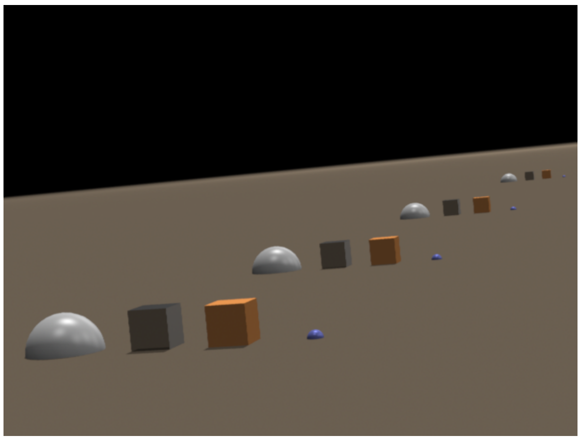

Figure 8: *Objects sliding down an inclined plane with friction. From left to right: analytic solution, rigid box, deformable box and a single particle.*

since these constraints are not decoupled. Furthermore, the residual only yields the net force, so we would not be able to untangle the effects of multiple interactions. Therefore, we propose a novel framework of incorporating the constraints into conjugate gradients only for the purpose of determining damping forces (and elastic forces if they are treated implicitly) and apply friction, contact, etc. as a postprocess. Step 3 uses the result of step 2 to compute the acceleration $\mathbf{a}$ as opposed to the projected acceleration $\hat{\mathbf{a}}$. Thus, all of the projections in step 2 only help to determine what velocity $\mathbf{v}_\star^{n+1}$ will be used to evaluate the damping forces in step 3. This approach produces accurate friction as illustrated in Figure 8.

An issue in our approach is that the postprocess of the velocities for constraint forces does not allow the damping forces to act on forces applied during the postprocess until the next time step. Thus, one needs to realize that there are undamped velocities, e.g. when computing a CFL condition. In addition, those velocities may get damped based on a smaller time step than the time step in which they were applied. We have nevertheless found that we can work around this by simply moving the postprocesses after applying explicit forces but before the conjugate gradient solve (the postprocesses still see explicit forces but not implicit forces and will thus be less accurate). We note that this was only done for the large pile example, and all other examples were run as described above.

7.3. Enforcing Constraints in the Solve

We enforce the linear constraints in the conjugate gradient solve in a momentum-conserving way. Let the constraints be written as $\mathbf{C}^T\mathbf{v} = 0$, where $\mathbf{C}$ is a matrix of coefficients and $\mathbf{v}$ is a vector containing the linear and angular velocities of all particles and rigid bodies. If M^{-1} is the block-diagonal mass matrix, the projection to be applied is $\delta - M^{-1}\mathbf{C}(\mathbf{C}^T M^{-1}\mathbf{C})^{-1}\mathbf{C}^T$, where δ is the identity matrix. This general form for the projection matrix holds for both rigid bodies and individual particles. When there are multiple constraints, we iterate these projections using forward and backward sweeps to ensure our projection step is symmetric even if it has not fully converged as in [ISF07]. The resulting matrix is always symmetric positive semidefinite and can be solved with conjugate gradients provided the residual is properly projected at each iteration.

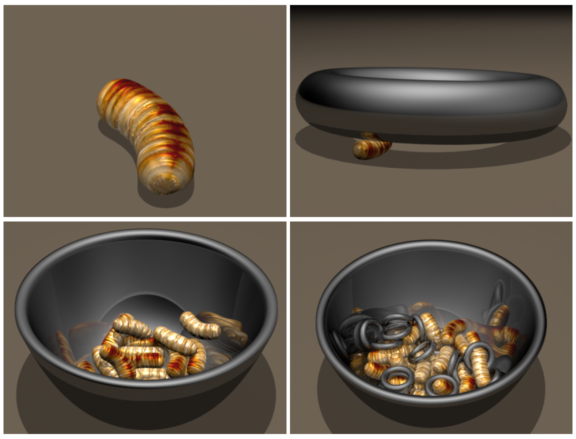

Figure 9: *Maggots constructed by embedding a two-joint PD-controlled articulated rigid body skeleton into a deformable body. The top row shows a maggot on the ground, and trapped under a large rigid body. The bottom row shows 20 maggots dropped into a bowl wriggling and interacting with each other, and the last image also has them interacting with 20 rigid tori.*

An alternative approach would be to solve the augmented system (or KKT system), which is symmetric and indefinite. This system was solved efficiently in [Bar96] for the case of explicitly integrated applied forces and a loop-free set of constraints. In the case of loops, such as those introduced by loops of articulated rigid bodies or implicitly integrated damping forces, this approach is no longer efficient. Another possible approach is to solve the KKT system with an iterative Krylov solver for indefinite systems, avoiding the projections at the cost of potentially inferior convergence characteristics.

Post-stabilization The post-stabilization algorithm of [WTF06] is included in our conjugate gradient solve to project the rigid body velocities in a momentum-conserving fashion so that they satisfy joint constraints. Since step 3 discards these velocities, they are reapplied in step 5.

Rigid/rigid contact We incorporate rigid/rigid contact constraints into the solve by creating joints just prior to the solve and removing them afterwards. In this fashion, we are able to use the articulated rigid body post-stabilization algorithm for projecting contact constraints during the conjugate gradient solve. When computing contact (during step 3 of Section 5) we record all points on the surface of one rigid body processed for contact against another. For each such point,

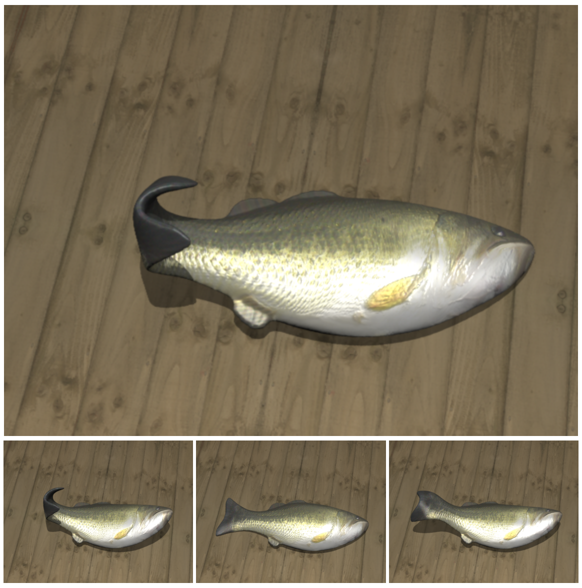

Figure 10: *A deformable fish constructed by embedding a four-joint PD-controlled articulated rigid body skeleton. The fish is dropped on the ground and flops back and forth interacting with its environment.*

we use its time t^{n+1} location and the level set normal from the other body at that point. We then construct a joint that only constrains motion in the normal direction and leaves the other two prismatic degrees of freedom and all angular degrees of freedom unconstrained.

Rigid/deformable contact Rigid/deformable contact projection is performed in a manner similar to interpenetration resolution as described in Section 6.1. We process a rigid body against all particles in contact with it simultaneously. However, we restrict impulses to the normal direction $\mathbf{n}_o$ during this contact processing phase. We target a relative change $\mathbf{v}_o$ that will cancel out the relative velocities in the normal direction of the particle and the body at the intersection point $\mathbf{p}_o$. If the central body is kinematic, the particles are given an impulse in the normal direction that cancels the relative normal velocity between the particle and the rigid body. Otherwise we apply impulses $j_o\mathbf{n}_o$ (where j_o is a scalar) to each outer body c at $\mathbf{p}_o$ so that

$$\mathbf{n}_o^T(m_c^{-1}\mathbf{j}+\mathbf{r}_o^{*T}I_c^{-1}\mathbf{j}_\tau - K_o j_o\mathbf{n}_o) = \mathbf{n}_o^T\mathbf{v}_o. \tag{3}$$

Using $L_o = \mathbf{n}_o\mathbf{n}_o^T(\mathbf{n}_o^T K_o\mathbf{n}_o)^{-1}$, we can express these equations in the form

$$\begin{pmatrix} m_c\delta+\sum\limits_o L_o & \sum\limits_o L_o\mathbf{r}_o^{*T} \\ \sum\limits_o \mathbf{r}_o^* L_o & I_c+\sum\limits_o \mathbf{r}_o^* L_o\mathbf{r}_o^{*T} \end{pmatrix}\begin{pmatrix} m_c^{-1}\mathbf{j} \\ I_c^{-1}\mathbf{j}_\tau \end{pmatrix} = \begin{pmatrix} -\sum\limits_o L_o\mathbf{v}_o \\ -\sum\limits_o \mathbf{r}_o^* L_o\mathbf{v}_o \end{pmatrix} \tag{4}$$

where the net linear and angular impulses applied to the central body are $\mathbf{j} = -\sum_o j_o\mathbf{n}_o$ and $\mathbf{j}_\tau = -\sum_o \mathbf{r}_o^* j_o\mathbf{n}_o$. This 6×6 system is symmetric positive definite and can be solved to obtain $\mathbf{j}$ and $\mathbf{j}_\tau$, from which j_o is obtained by substitution into Equation (3).

Self-repulsions Self-repulsions apply forces between edge-edge and point-triangle pairs to help avoid collisions [BFA02]. In the conjugate gradient solve, we use the projection algorithm proposed in [ISF07] to enforce these constraints.

8. Examples

Figure 5 illustrates that we can use the embeddings of [SSIF07b] in our fully two-way coupled approach. Note that [SSIF07b] was unable to handle two-way collisions and contact and more generally used an interleaved, semi-implicit approach to rigid/deformable coupling. Figure 1 demonstrates the robustness of the algorithm which allows stable two-way interaction between many types of competing constraints. Figure 4 demonstrates the ability of the two-way contact and collision algorithms to handle both volumetric objects and thin shells. Figure 6 demonstrates the ability of the algorithm to handle simulations with large numbers of two-way coupled bodies. One of the major applications of two-way rigid/deformable coupling is the simulation of skeleton-controlled deformable objects that can interact with their environment as shown in Figures 7, 9 and 10. Table 1 provides timing information for all of the examples in this paper. Most of the examples were fast enough that we simply ran them with conservative time steps, whereas the large pile was expensive enough to warrant performance tuning.

9. Conclusions

We propose a novel time integration scheme for two-way coupling rigid and deformable bodies that retains the strengths of deformable object simulators and rigid body simulators. We build upon existing algorithms for rigid/rigid and deformable/deformable interaction and where necessary propose new fully-coupled algorithms. The resulting scheme handles two-way coupled contact, collision, stacking, friction, articulation, and PD control. We use our framework to simulate life-like creatures that interact with their environment.

10. Acknowledgments

Research supported in part by a Packard Foundation Fellowship, an Okawa Foundation Research Grant, ONR N00014-06-1-0393, ONR N00014-06-1-0505, ONR N00014-02-1-0720, ONR N00014-05-1-0479 for a computing cluster, NIH U54-GM072970, NSF IIS-0326388, NSF ITR-0205671 and NSF CCF-0541148. We would also like to thank Sergey Levine for helping with examples.

References

[Bar96] BARAFF D.: Linear-time dynamics using Lagrange multipliers. In *Proc. of ACM SIGGRAPH 1996* (1996), pp. 137–146.

	Ave. Time Per Frame	Ave. Substeps Per Frame
Twisting Chain (Figure 1)	92.2 sec[†]	123
Stack (Figure 2)	10.0 sec	35
Row of Spheres (Figure 3)	35.3 sec	94
Deformable Bar (Figure 5)	4.1 sec	51
Trampoline (Figure 4)	14.5 sec	32
Large Pile (Figure 6)	40 min	14
Snake (Figure 7)	1.8 sec	38
Friction (Figure 8)	6.4 sec	2515[+]
Single Maggot (Figure 9)	0.6 sec	30
Maggot and Ring (Figure 9)	7.7 sec	100
Bowl of Maggots (Figure 9)	19.1 sec	30
...with Rings (Figure 9)	62.2 sec	59
Fish (Figure 10)	52.8 sec	117

Table 1: *This table contains average times per frame and number of substeps required per frame for each example.* [†] *The individual times varied substantially between frames, depending strongly on the degree of stress.* [+] *This test was run as a convergence test with an artificially low time step.*

[Ben07] BENDER J.: Impulse-based dynamic simulation in linear time. *Computer Animation and Virtual Worlds 18* (2007), 225–233.

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. 21*, 3 (2002), 594–603.

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* (2003), pp. 28–36.

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

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

[CDH06] CRISWELL B., DERLICH K., HATCH D.: Davy jones' beard: rigid tentacle simulation. In *SIGGRAPH 2006 Sketches* (2006), p. 117.

[FvT01] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Composable controllers for physics-based character animation. In *ACM Trans. Graph. (SIGGRAPH Proc.)* (2001), pp. 251–260.

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

[GHF*07] GOLDENTHAL R., HARMON D., FATTAL R., BERCOVIER M., GRINSPUN E.: Efficient simulation of inextensible cloth. *ACM Trans. Graph. 26*, 3 (2007), 49.

[GOT*07] GALOPPO N., OTADUY M., TEKIN S., GROSS M., LIN M. C.: Soft articulated characters with fast contact handling. *Comput. Graph. Forum (Proc. Eurographics) 26*, 3 (2007), 243–253.

[Hug00] HUGHES T. J. R.: *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover, 2000.

[HWBO95] HODGINS J., WOOTEN W., BROGAN D., O'BRIEN J.: Animating human athletics. In *Proc. of SIGGRAPH '95* (1995), pp. 71–78.

[ISF07] IRVING G., SCHROEDER C., FEDKIW R.: Volume conserving finite element simulations of deformable models. *ACM Trans. Graph. (SIGGRAPH Proc.) (in press) 26*, 3 (2007).

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

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

[MST*04] MÜLLER M., SCHIRM S., TESCHNER M., HEIDELBERGER B., GROSS M.: Interaction of fluids with deformable solids. *J. Comput. Anim. and Virt. Worlds 15*, 3–4 (July 2004), 159–171.

[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).

[Shi08] SHINAR T.: *Simulation of Coupled Rigid and Deformable Solids and Multiphase Fluids*. PhD thesis, Stanford University, June 2008.

[SSIF07a] SELLE A., SU J., IRVING G., FEDKIW R.: Highly detailed folds and wrinkles for cloth simulation. *IEEE Trans. on Vis. and Comput. Graph. (In Press)* (2007).

[SSIF07b] SIFAKIS E., SHINAR T., IRVING G., FEDKIW R.: Hybrid simulation of deformable solids. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* (2007), pp. 81–90.

[TSIF05] TERAN J., SIFAKIS E., IRVING G., FEDKIW R.: Robust quasistatic finite elements and flesh simulation. *Proc. of the 2005 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* (2005), 181–190.

[Wei07] WEINSTEIN R.: *Simulation and Control of Articulated Rigid Bodies*. PhD thesis, Stanford University, June 2007.

[WGF08] WEINSTEIN R., GUENDELMAN E., FEDKIW R.: Impulse-based control of joints and muscles. *IEEE Trans. on Vis. and Comput. Graph. 14*, 1 (2008), 37–46.

[WK88] WITKIN A., KASS M.: Spacetime constraints. In *Comput. Graph. (Proc. SIGGRAPH '88)* (1988), vol. 22, pp. 159–168.

[WTF06] WEINSTEIN R., TERAN J., FEDKIW R.: Dynamic simulation of articulated rigid bodies with contact and collision. *IEEE Trans. on Vis. and Comput. Graph. 12*, 3 (2006), 365–374.