Nonconvex Rigid Bodies with Stacking

Eran Guendelman* Stanford University
Robert Bridson* Stanford University
Ronald Fedkiw† Stanford University

Abstract

We consider the simulation of nonconvex rigid bodies focusing on interactions such as collision, contact, friction (kinetic, static, rolling and spinning) and stacking. We advocate representing the geometry with both a triangulated surface and a signed distance function defined on a grid, and this dual representation is shown to have many advantages. We propose a novel approach to time integration merging it with the collision and contact processing algorithms in a fashion that obviates the need for *ad hoc* threshold velocities. We show that this approach matches the theoretical solution for blocks sliding and stopping on inclined planes with friction. We also present a new shock propagation algorithm that allows for efficient use of the propagation (as opposed to the simultaneous) method for treating contact. These new techniques are demonstrated on a variety of problems ranging from simple test cases to stacking problems with as many as 1000 nonconvex rigid bodies with friction as shown in Figure 1.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: rigid bodies, collision, contact, friction, nonconvex

1 Introduction

Dynamic volumetric objects are pervasive in everyday life, and the ability to numerically simulate their behavior is important to a number of industries and applications including feature films, computer games and the automobile industry. One can differentiate between highly deformable volumetric objects and those where the deformation is either negligible or unimportant, and in the latter case efficiency concerns usually lead to rigid body approximations.

[Chatterjee and Ruina 1998] notes the weakness of the rigid body approximation to solids and discusses some known flaws in state of the art collision models. Moreover, they discuss some common misconceptions mentioning for example that the coefficient of restitution can be greater than one in frictional collisions. [Stewart 2000] emphasizes the difficulties with nonunique solutions pointing out that it is often impossible to predict which solution occurs in practice since it depends on unavailable details such as material microstructure. He states that one should repeat the calculations with random disturbances to characterize the potential set of solutions. [Barzel et al. 1996] exploited this indeterminacy by adding

*e-mail: {erang, rbridson}@stanford.edu
†e-mail: fedkiw@cs.stanford.edu

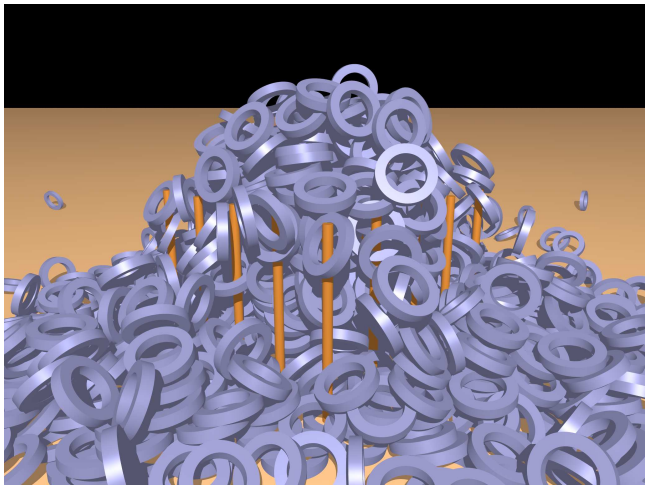

Figure 1: **1000 nonconvex rings (with friction) settling after being dropped onto a set of 25 poles. Each ring is made up of 320 triangles, and each pole is made up of 880 triangles.**

random texture and structured perturbations to enrich and control the motion respectively. The goal of rigid body simulation then becomes the construction of plausible motion instead of predictive motion. While the physicist strives towards synthesizing a family of solutions that are predictive in the sense that they statistically represent experimental data, the interest in graphics is more likely to focus on obtaining a particularly appealing solution from the set of plausible outcomes, e.g. see [Chenney and Forsyth 2000; Popović et al. 2000].

With this in mind, we focus on the plausible simulation of nonconvex rigid bodies emphasizing large scale problems with many frictional interactions. Although we start with a triangulated surface representation of the geometry, we also construct a signed distance function defined on a background grid (in the object frame) enabling the use of fast inside/outside tests. The signed distance function also conveniently provides a normal direction at points on and near the surface of the object. We take a closer look at the usual sequence of simulation steps—time integration, collision detection and modeling, contact resolution—and propose a novel approach that more cleanly separates collision from contact by merging both algorithms more tightly with the time integration scheme. This removes the need for *ad hoc* threshold velocities used by many authors to alleviate errors in the contact and collision algorithms, and correctly models difficult frictional effects without requiring micro-collision approximations [Mirtich and Canny 1995a; Mirtich and Canny 1995b] (which also use an *ad hoc* velocity). We also introduce a novel algorithm that increases the efficiency of the propagation method for contact resolution. The efficiency and robustness of our approach is illustrated with a number of simple and complex examples including frictional interactions and large contact groups as is typical in stacking.

2 Previous Work

[Hahn 1988] considered rigid body collisions by processing the collisions chronologically backing the rigid bodies up to the time of impact. [Mirtich 2000] used a timewarp algorithm to back up just

the objects that are involved in collisions while still evolving non-colliding objects forward in time. This method works well except when there are a large number of bodies in contact groups, which is the case we are concerned with in this paper. [Hahn 1988] processed collisions with static friction if the result was in the friction cone, and otherwise used kinetic friction. If the approach velocity was smaller than a threshold, the objects were assumed to be in contact and the same equations were applied approximating continuous contact with a series of "instantaneous contacts". [Moore and Wilhelms 1988] instead proposed the use of repulsion forces for contact only using the exact impulse-based treatment for high velocity collisions.

[Baraff 1989] proposed a method for analytically calculating non-colliding contact forces between polygonal objects obtaining an NP-hard quadratic programming problem which was solved using a heuristic approach. He also points out that these ideas could be useful in collision propagation problems such as one billiard ball hitting a number of others (that are lined up) or objects falling in a stack. [Baraff 1990] extended these concepts to curved surfaces. [Baraff 1991] advocated finding either a valid set of contact forces or a valid set of contact impulses stressing that the usual preference of the latter only when the former does not exist may be misplaced. For more details, see [Baraff 1993]. [Baraff 1994] proposed a simpler, faster and more robust algorithm for solving these types of problems without the use of numerical optimization software.

[Bhatt and Koechling 1995] discussed the nonlinear differential equations that need to be numerically integrated to analyze the behavior of three-dimensional frictional rigid body impact and pointed out that the problem becomes ill-conditioned at the sticking point. Then they use analysis to enumerate all the possible post-sticking scenarios and discuss the factors that determine a specific result. [Mirtich and Canny 1995b] integrated these same nonlinear differential equations to model both contact and collision proposing a unified model (as did [Hahn 1988]). They use the velocity an object at rest will obtain by falling through the collision envelope (or some threshold in [Mirtich and Canny 1995a]) to identify the contact case and apply a microcollision model that reverses the relative velocity as long as the required impulse lies in the friction cone. This solves the problem of blocks erroneously sliding down inclined planes due to impulse trains that cause them to spend time in a ballistic phase.

Implicitly defined surfaces were used for collision modeling by [Terzopoulos et al. 1987] to create repulsive force fields around objects and [Pentland and Williams 1989; Sclaroff and Pentland 1991] who exploited fast inside/outside tests. We use a particular implicit surface approach defining a signed distance function on an underlying grid. Grid-based distance functions have been gaining popularity, see e.g. [Fisher and Lin 2001; Hirota et al. 2001] who used them to treat collision between deformable bodies, and [Kim and Neumann 2002] who used them to keep hair from interpenetrating the head. Although [Gibson 1998] pointed out potential difficulties with spurious minima in concave regions, we have not noticed any adverse effects, most likely because we do not use repulsion forces.

[Milenkovic 1996] used position based physics to simulate the stacking of convex objects and discussed ways of making the simulations appear more physically realistic. [Milenkovic and Schmidl 2001] considered stacking with standard Newtonian physics using an optimization based method to adjust the predicted positions of the bodies to avoid overlap. One drawback is that the procedure tends to align bodies nonphysically. Quadratic programming is used for contact, collision and the position updates. They consider up to 1000 frictionless spheres, but nonconvex objects can only be considered as unions of convex objects and they indicate that the computational cost scales with the number of convex pieces. Their only nonconvex example considered 50 jacks that were each the union of 3 boxes. More recently, [Schmidl 2002] described a freezing technique that identifies when objects can be removed from the simulation, as well as identifying when to add them back. This allows the stacking of 1000 cubes with friction in 1.5 days as opposed to an estimated 45 days for simulating all the cubes.

3 Geometric Representation

Since rigid bodies do not deform, they are typically represented with triangulated surfaces, see Figure 2. Starting with either the density (or mass), algorithms such as [Mirtich 1996] can then be used to compute the volume, mass and moments of inertia. For efficiency, we store the object space representation with the center of mass at the origin and the axes aligned with the principal axes of inertia resulting in a diagonal inertia tensor simplifying many calculations, e.g. finding its inverse.

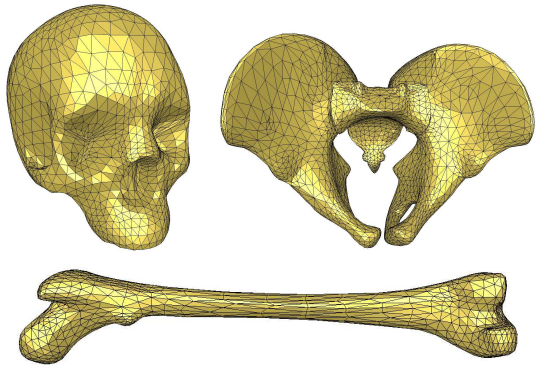

Figure 2: **Some nonconvex geometry from our simulations: the cranium, pelvis and femur have 3520, 8680 and 8160 triangles respectively.**

In addition to a triangulated surface, we also store an object space signed distance function for each rigid body. This is stored on either a uniform grid [Osher and Fedkiw 2002] or an octree grid [Frisken et al. 2000] depending on whether speed or memory, respectively, is deemed to be the bottleneck in the subsequent calculations. Discontinuities across octree levels are treated by constraining the fine grid nodes at gradation boundaries to take on the values dictated by interpolating from the coarse level, see e.g. [Westermann et al. 1999]. We use negative values of ϕ inside the rigid body and positive values of ϕ outside so that the normal is defined as $N = \nabla\phi$. This embedding provides approximations to the normal throughout space as opposed to just on the surface of the object allowing us to accelerate many contact and collision algorithms. A signed distance function can be calculated quickly using a marching method [Tsitsiklis 1995; Sethian 1996] after initializing grid points near the surface with appropriate small negative and positive values. This is a one time cost in constructing a rigid body model and is currently used in several systems, see e.g. [Cutler et al. 2002; Museth et al. 2002].

Using both a triangulated surface and a signed distance function representation has many advantages. For example, one can use the signed distance function to quickly check if a point is inside a rigid body, and if so intersect a ray in the $N = \nabla\phi$ direction with the triangulated surface to find the surface normal at the closest point. This allows the treatment of very sharp objects with their true surface normals, although signed distance function normals provide a smoother and less costly representation if desired. For more details on collisions involving sharp objects, see [Pandolfi et al. 2002].

4 Interference Detection

[Gascuel 1993; Desbrun and Gascuel 1995] found intersections between two implicitly defined surfaces by testing the sample points of one with the inside/outside function of the other. We follow the same strategy using the vertices of the triangulated surface as our

sample points. This test is not sufficient to detect all collisions, as edge-face collisions are missed when both edge vertices are outside the implicit surface. Since the errors are proportional to the edge length, they can be ignored in a well resolved mesh with small triangles. However, when substantial, e.g. when simulating cubes with only 12 triangles, we intersect the triangle edges with the zero isocontour and flag the deepest point on the edge as an interpenetrating sample point. Since we do not consider time dependent collisions, fast moving objects might pass through each other. We alleviate this problem by limiting the size of a time step based on the translational and rotational velocities of the objects and the size of their bounding boxes, although methods exist for treating the entire time swept path as a single implicit surface [Schroeder et al. 1994].

A number of accelerations can be used in the interference detection process. For example, the inside/outside tests can be accelerated by labeling the voxels that are completely inside and completely outside (this is done for voxels at each level in the octree representation as well) so that interpolation can be avoided except in cells which contain part of the interface. Labeling the minimum and maximum values of ϕ in each voxel can also be useful. Bounding boxes and spheres are used around each object in order to prune points before doing a full inside/outside test. Moreover, if the bounding volumes are disjoint, no inside/outside tests are needed. For rigid bodies with a large number of triangles, we found an internal box hierarchy with triangles in leaf boxes to be useful especially when doing edge intersection tests. Also, we use a uniform spatial partitioning data structure with local memory storage implemented using a hash table in order to quickly narrow down which rigid bodies might be intersecting. Similar spatial partitioning was used in, for example, [Mirtich and Canny 1995b]. Again, we stress our interest in nonconvex objects and refer the reader to [Ponamgi et al. 1995; Kim et al. 2002] for other algorithms that treat arbitrary nonconvex polyhedral models. For more details on collision detection methods, see e.g. [Webb and Gigante 1992; Lin and Gottschalk 1998; Redon et al. 2002].

5 Time Integration

The equations for rigid body evolution are

$$x_t = v, \quad q_t = \tfrac{1}{2}\omega q \tag{1}$$

$$v_t = F/m, \quad L_t = \tau \tag{2}$$

where x and q are the position and orientation (a unit quaternion), v and ω are the velocity and angular velocity, F is the net force, m is the mass, $L = I\omega$ is the angular momentum with inertia tensor $I = RDR^T$ (R is the orientation matrix and D is the diagonal inertia tensor in object space), and τ is the net torque. For simplicity we will consider $F = mg$ and thus $v_t = g$ throughout the text, but our algorithm is not restricted to this case. While there are a number of highly accurate time integration methods for noninteracting rigid bodies in free flight, see e.g. [Buss 2000], these algorithms do not retain this accuracy in the presence of contact and collision. Thus, we take a different approach to time integration instead optimizing the treatment of contact and collision. Moreover, we use a simple forward Euler time integration for equations 1 and 2.

The standard approach is to integrate equations 1 and 2 forward in time, and subsequently treat collision and then contact. Generally speaking, collisions require impulses that discontinuously modify the velocity, and contacts are associated with forces and accelerations. However, friction can require the use of impulsive forces in the contact treatment, although the *principle of constraints* requires that the use of impulsive forces be kept to a minimum. [Baraff 1991] suggested that this avoidance of impulsive behavior is neither necessary nor justified and stressed that there are algorithmic advantages to using impulses exclusively. This naturally leads to some blurring between collision and contact handling, and provides a sense of justification to the work of [Hahn 1988] where the same algebraic equations were used for both and the work of [Mirtich and Canny 1995b] who integrated the same nonlinear differential equations for both. However, other authors such as [Moore and Wilhelms 1988; Sims 1994; Kokkevis et al. 1996] have noted difficulties associated with this blurring and proposed that an impulse based treatment of collisions be separated from a penalty springs approach to contact. They used the magnitude of the relative velocity to differentiate between contact and collision. [Mirtich and Canny 1995b] used the velocity an object at rest will obtain by falling through the collision envelope (or some threshold in [Mirtich and Canny 1995a]) to identify the contact case and applied a microcollision model where the impulse needed to reverse the relative velocity is applied as long as it lies in the friction cone. They showed that this solves the problem of blocks erroneously sliding down inclined planes due to impulse trains that cause them to spend time in a ballistic phase.

A novel aspect of our approach is the clean separation of collision from contact without the need for threshold velocities. We propose the following time sequencing:

- Collision detection and modeling.
- Advance the velocities using equation 2.
- Contact resolution.
- Advance the positions using equation 1.

The advantages of this time stepping scheme are best realized through an example. Consider a block sitting still on an inclined plane with a large coefficient of restitution, say $\varepsilon = 1$, and suppose that friction is large enough that the block should sit still. In a standard time stepping scheme, both position and velocity are updated first, followed by collision and contact resolution. During the position and velocity update, the block starts to fall under the effects of gravity. Then in the collision processing stage we detect a low velocity collision between the block and the plane, and since $\varepsilon = 1$ the block will change direction and bounce upwards at an angle down the incline. Then in the contact resolution stage, the block and the plane are separating so nothing happens. The block will eventually fall back to the inclined plane, and continue bouncing up and down incorrectly sliding down the inclined plane because of the time it spends in the ballistic phase. This is the same phenomenon that causes objects sitting on the ground to vibrate as they are incorrectly subjected to a number of elastic collisions. Thus, many authors use *ad hoc* threshold velocities in an attempt to prune these cases out of the collision modeling algorithm and instead treat them with a contact model.

Our new time stepping algorithm automatically treats these cases. All objects at rest have zero velocities (up to round-off error), so in the collision processing stage we do not get an elastic bounce (up to round-off error). Next, gravity is integrated into the velocity, and then the contact resolution algorithm correctly stops the objects so that they remain still. Thus, nothing happens in the last (position update) step, and we repeat the process. The key to the algorithm is that contact modeling occurs directly after the velocity is updated with gravity. If instead either the collision step or a position update were to follow the velocity update, objects at rest will either incorrectly elastically bounce or move through the floor, respectively. On the other hand, contact processing is the correct algorithm to apply after the velocity update since it resolves forces, and the velocity update is where the forces are included in the dynamics.

One must use care when updating the velocity in between the collision and contact algorithms to ensure that the same exact technique is used to detect contact as was used to detect collision. Otherwise, an object in free flight might not register a collision, have its velocity updated, and then register a contact causing it to incorrectly receive an inelastic (instead of elastic) bounce. We avoid this situation by guaranteeing that the contact detection step registers a negative result whenever the collision detection step does. This is

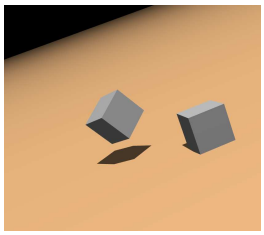

Figure 3: **The block and inclined plane test with standard time integration (the block erroneously tumbling) and our new time integration sequencing (the block correctly at rest).**

easily accomplished by ensuring that the velocity update has no effect on the contact and collision detection algorithms (discussed in Section 6).

We repeated the experiment of a block sliding down an inclined plane from [Mirtich and Canny 1995b] using the methods for collision and contact proposed throughout this paper and our newly proposed time step sequencing. We used a coefficient of restitution $\varepsilon = 1$ in order to accentuate difficulties with erroneous elastic bouncing. Using our new time stepping scheme the decelerating block slides down the inclined plane coming to a stop matching theory, while the standard time stepping scheme performs so poorly that the block bounces down the inclined plane as shown in Figure 3. Of course, these poor results are accentuated because we both set $\varepsilon = 1$ and do not back up the simulation to the time of collision (which would be impractical and impossible for our large stacking examples). Figure 4 shows a comparison between theory and our numerical results. For both the acceleration and deceleration cases, our numerical solution and the theoretical solution lie so closely on top of each other that two distinct lines cannot be seen. Moreover, our results are noticeably better than those depicted in [Mirtich and Canny 1995b] even though we do not use a threshold velocity or their microcollision model.

6 Collisions

When there are many interacting bodies, it can be difficult to treat all the collisions especially if they must be resolved in chronological order. Thus instead of rewinding the simulation to process collisions one at a time, we propose a method that simultaneously resolves collisions as did [Stewart and Trinkle 2000; Milenkovic and Schmidl 2001]. While this does not give the same result as processing the collisions in chronological order, there is enough uncertainty in the collision modeling that we are already guaranteed to not get *the* exact physically correct answer. Instead we will obtain *a* physically plausible solution, i.e. one of many possible physically correct outcomes which may vary significantly with slight perturbations in initial conditions or the inclusion of unmodeled phenomena such as material microstructure.

Collisions are detected by predicting where the objects will move to in the next time step, temporarily moving them there, and checking for interference. The same technique will be used for detecting contacts, and we want the objects to be moved to the same position for both detection algorithms if there are no collisions (as mentioned above). In order to guarantee this, we use the new velocities to predict the positions of the rigid bodies in both steps. Of course, we still use the old velocities to process the collisions and the new velocities to process the contacts. For example, for the collision phase, if an object's current position and velocity are x and v, we test for interference using the predicted position $x' = x + \Delta t(v + \Delta t g)$, and apply collision impulses to (and using) the current velocity v. During contact processing, we use the predicted position $x' = x + \Delta t v'$ and apply impulses to this new velocity v'. Since $v' = v + \Delta t g$ was set in the velocity update step, the candidate positions match and the interference checks are consistent.

The overall structure of the algorithm consists of first moving

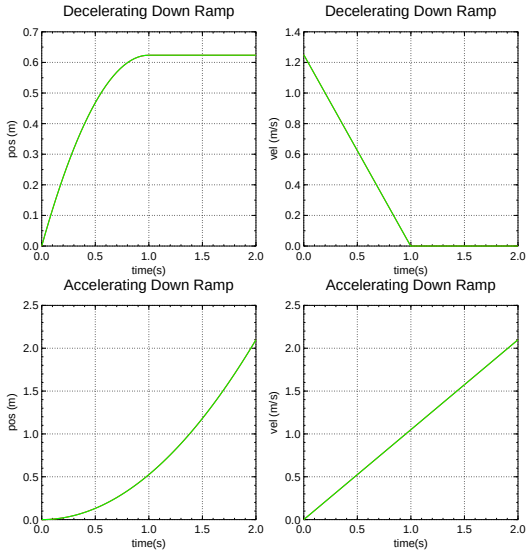

Figure 4: **Theoretical and our numerical results for two tests of a block sliding down an inclined plane with friction. The curves lie on top of each other in the figures due to the accuracy of our new time sequencing algorithm.**

all rigid bodies to their predicted locations, and then identifying and processing all intersecting pairs. Since collisions change the rigid body's velocity, v, new collisions may occur between pairs of bodies that were not originally identified. Therefore we repeat the entire process a number of times (e.g. five iterations) moving objects to their newly predicted locations and identifying and processing all intersecting pairs. Since pairs are considered one at a time, the order in which this is done needs to be determined. This can be accomplished by initially putting all the rigid bodies into a list, and then considering rigid bodies in the order in which they appear. To reduce the inherent bias in this ordering, we regularly mix up this list by randomly swapping bodies two at a time. This list is used throughout our simulation whenever an algorithm requires an ordering.

For each intersecting pair, we identify all the vertices of each body that are inside the other (and optionally the deepest points on interpenetrating edges as well). Since we do not back up the rigid bodies to the time of collision, we need a method that can deal with nonconvex objects with multiple collision regions and multiple interfering points in each region. We start with the deepest point of interpenetration that has a nonseparating relative velocity as did [Moore and Wilhelms 1988], and use the standard algebraic collision laws (below) to process the collision. Depending on the magnitude of the collision, this may cause the separation of the entire contact region. If we re-evolve the position using the new post-collision velocity, this collision group could be resolved. Whether or not it is resolved, we can once again find the deepest non-separating point and repeat the process until all points are either non-interpenetrating or separating. While this point sampling method is not as accurate as integrating over the collision region as in [Hirota et al. 2001], it is much faster and scales well to large numbers of objects.

We developed an aggressive optimization for the point sampling that gives similarly plausible results, (see e.g. Figure 5). As before, one first labels all the non-separating intersecting points and applies a collision to the deepest point. But instead of re-evolving the objects and repeating the expensive collision detection algorithm, we simply keep the objects stationary and use the same list of (initially) interfering points for the entire procedure. After processing the collision, all separating points are removed from the list. Then the remaining deepest nonseparating point is identified and the process is repeated until the list is empty. In this manner, all points in the original list are processed until they are separating *at least once*

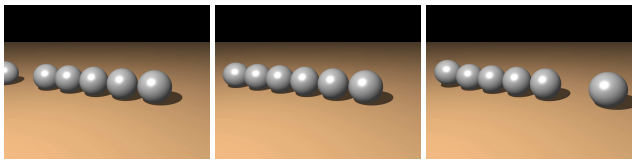

Figure 5: **A billiard ball hits one end of a line of billiard balls, and the collision response propagates to the ball on the far right which then starts to roll.**

during the procedure. The idea of lagging collision geometry was also considered by [Baraff 1995] in a slightly different context.

Each body is assigned a coefficient of restitution, and when two bodies collide we use the minimum between the two coefficients as did [Moore and Wilhelms 1988] to process the collision. Suppose the relative velocity at the collision point was originally u_{rel} with (scalar) normal and (vector) tangential components, $u_{rel,n} = u_{rel} \cdot N$ and $u_{rel,t} = u_{rel} - u_{rel,n}N$ respectively. Then we apply equal and opposite impulses j to each body to obtain $v' = v \pm j/m$ and $\omega' = \omega \pm I^{-1}(r \times j)$ where r points from their respective centers of mass to the collision location. The new velocities at the point of collision will be $u' = u \pm Kj$ where $K = \delta/m + r^{*T} I^{-1} r^*$ with δ the identity matrix and the "$*$" superscript indicating the cross-product matrix. Finally, $u'_{rel,n} = u_{rel,n} + N^T K_T N j_n$ where K_T is the sum of the individual K's and $j = j_n N$ is our frictionless impulse. So given a final relative normal velocity $u'_{rel,n} = -\varepsilon u_{rel,n}$, we can find the impulse j. Immovable static objects like the ground plane can be treated by setting $K = 0$ and not updating their velocities.

7 Static and Kinetic Friction

The collision algorithm above needs to be modified to account for kinetic and static friction. Each body is assigned a coefficient of friction, and we use the maximum of the two possible coefficients when processing a collision as did [Moore and Wilhelms 1988]. Like [Hahn 1988; Moore and Wilhelms 1988], we first assume that the bodies are stuck at the point of impact due to static friction and solve for the impulse. That is, we set $u'_{rel,t} = 0$ so that $u'_{rel} = -\varepsilon u_{rel,n} N$ allows us to solve $u'_{rel} = u_{rel} + K_T j$ for the impulse j by inverting the symmetric positive definite matrix K_T. Then if j is in the friction cone, i.e. if $|j - (j \cdot N)N| \leq \mu j \cdot N$, the point is sticking due to static friction and j is an acceptable impulse. Otherwise, we apply sliding friction.

Define $T = u_{rel,t}/|u_{rel,t}|$ so that the kinetic friction can be computed with the impulse $j = j_n N - \mu j_n T$. Then take the dot product of $u'_{rel} = u_{rel} + K_T j$ with N to obtain $u'_{rel,n} = u_{rel,n} + N^T K_T j$ or $-\varepsilon u_{rel,n} = u_{rel,n} + N^T K_T j$. Plugging in the definition of j we can solve to find $j_n = -(\varepsilon+1)u_{rel,n}/(N^T K_T (N - \mu T))$ from which the kinetic friction impulse j is determined.

8 Contact

After a few iterations of the collision processing algorithm, the rigid bodies have been elastically bounced around enough to obtain a plausible behavior. So even if collisions are still occurring, we update the velocities of all the rigid bodies and move on to contact resolution. Since the contact modeling algorithm is similar to the collision modeling algorithm except with a zero coefficient of restitution, objects still undergoing collision will be processed with inelastic collisions. This behavior is plausible since objects undergoing many collisions in a single time step will tend to rattle around and quickly lose energy.

The goal of the contact processing algorithm is to resolve the forces between objects. As in collision detection, we detect contacts by predicting where the objects will move to in the next time step disregarding the contact forces, temporarily moving them there, and checking for interference. For example, objects sitting on the ground will fall into the ground under the influence of gravity leading to the flagging of these objects for contact resolution. All interacting pairs are flagged and processed in the order determined by our list. Once again, multiple iterations are needed especially for rigid bodies that sit on top of other rigid bodies. For example, a stack of cubes will all fall at the same speed under gravity and only the cube on the bottom of the stack will intersect the ground and be flagged for contact resolution. The other cubes experience no interference in this first step. However, after processing the forces on the cube at the bottom of the stack, it will remain stationary and be flagged as interpenetrating with the cube that sits on top of it in the next sweep of the algorithm. This is a propagation model for contact as opposed to the simultaneous solution proposed in [Baraff 1989].

The difficulty with a propagation model is that it can take many iterations to converge. For example, in the next iteration the cube on the ground is stationary and the cube above it is falling due to gravity. If we process an inelastic collision between the two cubes, the result has both cubes falling at half the speed that the top cube was falling. That is, the cube on top does not stop, but only slows down. Even worse, the cube on the ground is now moving again and we have to reprocess the contact with the ground to stop it. In this sense, many iterations are needed since the algorithm does not have a global view of the situation. That is, all the non-interpenetration constraints at contacts can be viewed as one large system of equations, and processing them one at a time is similar to a slow Gauss-Seidel approach to solving this system. Instead, if we simultaneously considered the entire system of equations, one could hope for a more efficient solution, for example by using a better iterative solver. This is the theme in [Milenkovic and Schmidl 2001] where an optimization based approach is taken. We propose a more light-weight method in Section 8.2.

Similar to the collision detection algorithm, for each intersecting pair, we identify all the vertices of each body that are inside the other (and optionally the deepest points on edges as well). Although [Baraff 1989] pointed out that the vertices of the contact region (which lie on the vertices and edges of the original model) need to be considered, we have found our point sampling method to be satisfactory. However, since we have a triangulated surface for each object, we could do this if necessary. As in [Hahn 1988; Mirtich and Canny 1995b] we use the same equations to process each contact impulse that were used in the collision algorithm, except that we set $\varepsilon = 0$. We start with the deepest point of interpenetration that has a non-separating relative velocity, and again use the standard algebraic collision laws. Then a new predicted position can be determined and the process repeated until all points are either non-overlapping or separating. Although the aggressive optimization algorithm that processes all points until they are separating *at least once* could be applied here as well, it is not as attractive for contact as it is for collision since greater accuracy is usually desired for contacts.

For improved accuracy, we propose the following procedure. Rather than applying a fully inelastic impulse of $\varepsilon = 0$ at each point of contact, we *gradually* stop the object from approaching. For example, on the first iteration of contact processing we apply impulses using $\varepsilon = -.9$, on the next iteration we use $\varepsilon = -.8$, and so on until we finally use $\varepsilon = 0$ on the last iteration. A negative coefficient of restitution simply indicates that rather than stop or reverse an approaching object, we only slow it down.

In the collision processing algorithm, we used the predicted positions to determine the geometry (e.g. normal) of the collision. Although it would have been better to use the real geometry at the time of collision, the collision time is not readily available and furthermore the accuracy is not required since objects are simply bouncing around. On the other hand, objects should be sitting still in the contact case, and thus more accuracy is required to prevent incorrect

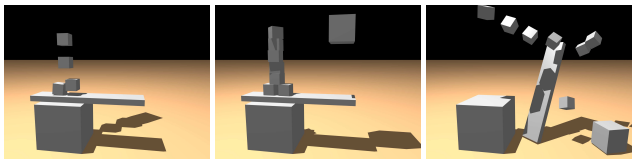

Figure 6: **Although the propagation treatment of contact and collision allows the stacking and flipping of boxes as shown in the figure, our shock propagation algorithm makes this both efficient and visually accurate.**

rattling around of objects. Moreover, the correct contact geometry is exactly the current position (as opposed to the predicted position), since the contact forces should be applied before the object moved. Thus, we use the current position to process contacts.

8.1 Contact graph

At the beginning of the contact resolution stage we construct a contact graph similar to [Hahn 1988; Baciu and Wong 2000] with the intention of identifying which bodies or groups of bodies are resting on top of others. We individually allow each object to fall under the influence of gravity (keeping the others stationary) for a characteristic time $\triangle\tau$ (on the order of a time step), and record all resulting interferences adding a directed edge pointing towards the falling object from the other object. Then we apply a simple topological sort algorithm that uses two depth first searches to collapse strongly connected components resulting in a directed acyclic graph. For a stack of cubes, we get a contact graph that points from the ground up one cube at a time to the top of the stack. For difficult problems such as a set of dominoes arranged in a circle on the ground with each one resting on top of the one in front of it, we simply get the ground in one level of the contact graph and *all* the dominoes in a second level. Roughly speaking, objects are grouped into the same level if they have a cyclic dependence on each other.

The purpose of the contact graph is to suggest an order in which contacts should be processed, and we wish to sweep up and out from the ground and other static (non-simulated) objects in order to increase the efficiency of the contact propagation model. When considering objects in level i, we gather all contacts between objects within level i as well as contacts between objects in this level and ones at lower-numbered levels. With the latter type of contact pairs, the object in level i is, as a result of the way we constructed the contact graph, necessarily "resting on" the lower level object and not the other way around. The contact pairs found for level i are put into a list and treated in any order iterating through this list a number of times before moving on to the next level. Additionally, we sweep along the graph through all levels multiple times for improved accuracy.

8.2 Shock propagation

Even with the aid of a contact graph, the propagation model for contact may require many iterations to produce visually appealing results especially in simulations with stacks of rigid bodies. For example, in the cube stack shown in Figure 6 (center), the cubes will start sinking into each other if not enough iterations are used. To alleviate this effect, we propose a *shock propagation* method that can be applied on the last sweep through the contact graph. After each level is processed in this last sweep, all the objects in that level are assigned infinite mass (their K matrix is set to zero). Here, the benefit of sorting the objects into levels becomes most evident. If an object of infinite mass is later found to be in contact with a higher-level object, its motion is not affected by the impulses applied to resolve contact, and the higher level object will simply have to move out of the way! Once assigned infinite mass, objects retain this mass until the shock propagation phase has completed. As in contact, we iterate a number of times over all contact pairs in each level, but unlike contact we only complete one sweep through all of the levels. Note that when two objects at the same level are in contact, neither has been set to infinite mass yet, so shock propagation in this case is no different than our usual contact processing. However, the potentially slow convergence of the usual contact processing has now been localized to the smaller groups of strongly connected components in the scene.

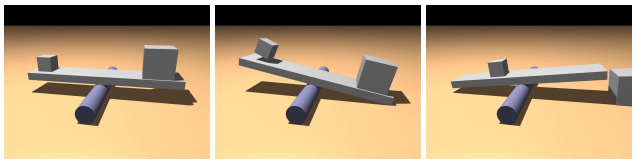

Figure 7: **A heavier block on the right tips the see-saw in that direction, and subsequently slides off. Then the smaller block tips the see-saw back in the other direction. The propagation treatment of contact allows the weight of each block to be felt, and our shock propagation method keeps the blocks from interpenetrating without requiring a large number of contact processing iterations.**

To see how this algorithm works, consider the stack of objects in Figure 6 (center). Starting at the bottom of the stack, each object has its velocity set to zero and its mass subsequently set to be infinite. As we work our way up the stack, the falling objects cannot push the infinite mass objects out of the way so they simply get their velocity set to zero as well. In this fashion, the contact graph allows us to shock the stack to a zero velocity in one sweep.

In order to demonstrate why the propagation model for contact is used for a few iterations before applying shock propagation we drop a larger box onto the plank as shown in Figure 6 (center). Here the contact graph points up from the ground through all the objects, and when the larger box first comes in contact with the plank, an edge will be added pointing from the plank to the box. If shock propagation was applied immediately the box on the ground and then the plank would have infinite mass. Thus the large falling box would simply see an infinite mass plank and be unable to flip over the stack of boxes as shown in Figure 6 (center). However, contact propagation allows both the plank to push up on the falling box *and* the falling box to push back down. That is, even though "pushing down" increases the number of iterations needed for the contact algorithm to converge, without this objects would not feel the weight of other objects sitting on top of them. Thus, we sweep though our contact graph a number of times in order to get a sense of weight, and then efficiently force the algorithm to converge with a final shock propagation sweep. This allows the stack of boxes to flip over as shown in Figure 6 (right).

Figure 7 shows another test where a heavy and a light block are both initially at rest on top of a see-saw. When the simulation starts the weight of the heavier block pushes down tilting the see-saw in that direction. Eventually it tilts enough for the heavy block to slide off, and then the see-saw tilts back in the other direction under the weight of the lighter block. Our combination of contact propagation followed by shock propagation correctly and efficiently handles this scenario. On the other hand, if we run shock propagation only (i.e. omitting the contact propagation phase), the see-saw either sits still or tips very slowly since it does not feel the weight of the heavy block.

9 Rolling and Spinning Friction

Even when a rigid body has a contact point frozen under the effects of static friction, it still has freedom to both roll and spin. [Lewis and Murray 1995; Sauer and Schömer 1998] damped these degrees of freedom by adding forces to emulate rolling friction and air drag. Instead, we propose an approach that treats these effects in the same manner as kinetic and static friction. Let μ_r and μ_s designate the coefficients of rolling and spinning friction, and note that

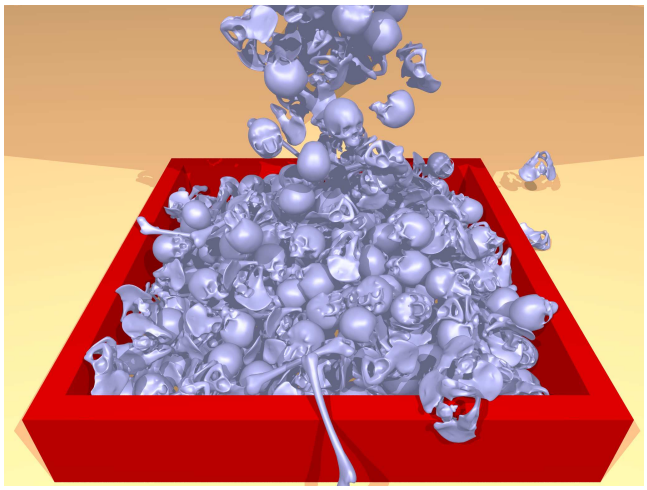

Figure 8: **500 nonconvex bones falling into a pile after passing through a funnel. Exact triangle counts are given in figure 2.**

these coefficients should depend on the local deformation of the object. This means that they should by scaled by the local curvature with higher values in areas of lower curvature. Thus for a sphere, these values are constant throughout the object.

Both rolling and spinning friction are based on the relative angular velocity ω_{rel} with normal and tangential components $\omega_{rel,n} = \omega_{rel} \cdot N$ and $\omega_{rel,t} = \omega_{rel} - \omega_{rel,n}N$. The normal component governs spinning and the tangential component governs rolling about $T = \omega_{rel,t}/|\omega_{rel,t}|$. We modify these by reducing the magnitude of the normal and tangential components by $\mu_s j_n$ and $\mu_r j_n$ respectively. To keep the object from reversing either its spin or roll direction, both of these reductions are limited to zero otherwise preserving the sign. At this point we have a new relative angular velocity ω'_{rel}, and since the objects are sticking the relative velocity at the contact point is $u'_{rel} = 0$. Next, we construct an impulse to achieve both proposed velocities.

If we apply the impulse at a point, specifying the relative velocity determines the impulse j and we are unable to also specify the relative angular velocity. Thus, we assume that the impulse is applied over an area instead. We still have $v' = v \pm j/m$ for the center of mass velocity, but the angular velocity is treated differently. First we explicitly write out the change in angular momentum (about our fixed world origin) due to an angular impulse j_τ as $x \times mv' + I\omega' = x \times mv + I\omega \pm j_\tau$ which can be rearranged to give $\omega' = \omega \pm I^{-1}(j_\tau - x \times j)$. When j_τ is equal to the cross product of the point of contact and j, this reduces to $\omega' = \omega \pm I^{-1}(r \times j)$ as above, but here we consider the more general case in order to get control over the angular velocity. The new velocities at the point of collision will be $u' = u \pm (K_1 j + K_2 j_\tau)$ where $K_1 = \delta/m + r^* I^{-1} x^*$ and $K_2 = -r^* I^{-1}$. Then $u'_{rel} = u_{rel} + K_{1,T} j + K_{2,T} j_\tau$ where the $K_{i,T}$'s are the sum of the individual K_i's. Similar manipulation gives $\omega'_{rel} = \omega_{rel} + K_{3,T} j + K_{4,T} j_\tau$, where $K_3 = -I^{-1} x^*$ and $K_4 = I^{-1}$. This is two equations in two unknowns, j and j_τ, so we can solve one equation for j, plug it into the other, and solve for j_τ. This requires inverting two 3×3 matrices.

10 Results

Besides the basic tests that we discussed throughout the text, we also explored the scalability of our algorithm addressing simulations of large numbers of nonconvex objects with high resolution triangulated surfaces falling into stacks with multiple contact points. While the CPU times for the simple examples were negligible, the simulations depicted in Figures 1, 8 and 9 had a significant computational cost. Dropping 500 and 1000 rings into a stack averaged about 3 minutes and 7 minutes per frame, respectively.

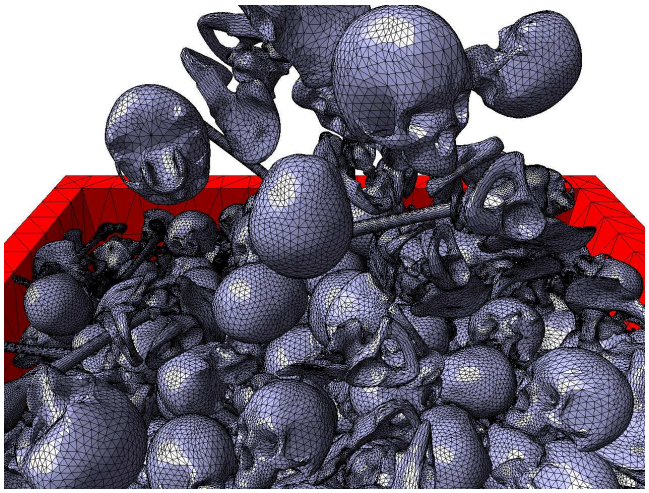

Figure 9: **The complexity of our nonconvex objects is emphasized by exposing their underlying tessellation.**

Dropping 500 bones through a funnel into a pile averaged about 7 minutes per frame, and we note that this simulation had about 2.8 million triangles total. All examples were run using 5 collision iterations, 10 contact iterations, and a single shock propagation iteration, and all used friction.

11 Conclusions and Future Work

We proposed a mixed representation of the geometry combining triangulated surfaces with signed distance functions defined on grids and illustrated that this approach has a number of advantages. We also proposed a novel time integration scheme that removes the need for *ad hoc* threshold velocities and matches theoretical solutions for blocks sliding and stopping on inclined planes. Finally, we proposed a new shock propagation method that dramatically increases the efficiency and visual accuracy of stacking objects. So far, our rolling and spinning friction model has only produced good results for spheres and we are currently investigating more complex objects.

After the positions have been updated, interpenetration may still occur due to round-off errors and in-level contact between objects. We experimented with the use of first order physics (similar to [Baraff 1995]) to compute a "first order impulse" to apply to the objects' positions and orientations to effect separation (without modifying their velocities). As in shock propagation, we proceeded level by level through the contact graph doing multiple iterations of separation adjustments within each level before assigning infinite masses to each level in preparation for the next. To reduce bias, we *gradually* separated objects within a level, each iteration increasing the fraction of interpenetration that is corrected. We plan to investigate this further in future work.

12 Acknowledgements

Research supported in part by an ONR YIP and PECASE award (ONR N00014-01-1-0620), a Packard Foundation Fellowship, a Sloan Research Fellowship, ONR N00014-03-1-0071, ONR N00014-02-1-0720, NSF ITR-0121288 and NSF ACI-0205671. In addition, R. B. was supported in part by a Stanford Graduate Fellowship.

References

BACIU, G., AND WONG, S. K. 2000. The impulse graph: a new dynamic structure for global collisions. *CGForum 19*, 3, 229–238.

BARAFF, D. 1989. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Comput. Graph. (Proc. SIGGRAPH 89) 23*, 3, 223–232.

BARAFF, D. 1990. Curved surfaces and coherence for non-penetrating rigid body simulation. *Comput. Graph. (Proc. SIGGRAPH 90) 24*, 4, 19–28.

BARAFF, D. 1991. Coping with friction for non-penetrating rigid body simulation. *Comput. Graph. (Proc. SIGGRAPH 91) 25*, 4, 31–40.

BARAFF, D. 1993. Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica*, 10, 292–352.

BARAFF, D. 1994. Fast contact force computation for nonpenetrating rigid bodies. In *Proc. SIGGRAPH 94*, 23–34.

BARAFF, D. 1995. Interactive simulation of solid rigid bodies. *IEEE Comput. Graph. and Appl. 15*, 3, 63–75.

BARZEL, R., HUGHES, J. F., AND WOOD, D. N. 1996. Plausible motion simulation for computer graphics. In *Comput. Anim. and Sim. '96*, Proc. Eurographics Workshop, 183–197.

BHATT, V., AND KOECHLING, J. 1995. Three-dimensional frictional rigid-body impact. *ASME J. Appl. Mech. 62*, 893–898.

BUSS, S. R. 2000. Accurate and efficient simulation of rigid-body rotations. *J. Comput. Phys. 164*, 2.

CHATTERJEE, A., AND RUINA, A. 1998. A new algebraic rigid body collision law based on impulse space considerations. *ASME J. Appl. Mech. 65*, 4, 939–951.

CHENNEY, S., AND FORSYTH, D. A. 2000. Sampling plausible solutions to multi-body constraint problems. In *Proc. SIGGRAPH 2000*, 219–228.

CUTLER, B., DORSEY, J., MCMILLAN, L., MÜLLER, M., AND JAGNOW, R. 2002. A procedural approach to authoring solid models. *ACM Trans. Graph. (SIGGRAPH Proc.) 21*, 3, 302–311.

DESBRUN, M., AND GASCUEL, M.-P. 1995. Animating soft substances with implicit surfaces. In *Proc. SIGGRAPH 95*, 287–290.

FISHER, S., AND LIN, M. C. 2001. Deformed distance fields for simulation of non-penetrating flexible bodies. In *Comput. Anim. and Sim. '01*, Proc. Eurographics Workshop, 99–111.

FRISKEN, S. F., PERRY, R. N., ROCKWOOD, A. P., AND JONES, T. R. 2000. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *Proc. SIGGRAPH 2000*, 249–254.

GASCUEL, M.-P. 1993. An implicit formulation for precise contact modeling between flexible solids. In *Proc. SIGGRAPH 93*, 313–320.

GIBSON, S. F. F. 1998. Using distance maps for accurate surface representation in sampled volumes. In *Proc. of IEEE Symp. on Vol. Vis.*, 23–30.

HAHN, J. K. 1988. Realistic animation of rigid bodies. *Comput. Graph. (Proc. SIGGRAPH 88) 22*, 4, 299–308.

HIROTA, G., FISHER, S., STATE, A., LEE, C., AND FUCHS, H. 2001. An implicit finite element method for elastic solids in contact. In *Comput. Anim.*

KIM, T.-Y., AND NEUMANN, U. 2002. Interactive multiresolution hair modeling and editing. *ACM Trans. Graph. (SIGGRAPH Proc.) 21*, 3, 620–629.

KIM, Y. J., OTADUY, M. A., LIN, M. C., AND MANOCHA, D. 2002. Fast penetration depth computation for physically-based animation. In *ACM Symp. Comp. Anim.*

KOKKEVIS, E., METAXAS, D., AND BADLER, N. 1996. User-controlled physics-based animation for articulated figures. In *Proc. Comput. Anim. '96*.

LEWIS, A. D., AND MURRAY, R. M. 1995. Variational principles in constrained systems: theory and experiments. *Int. J. Nonlinear Mech. 30*, 6, 793–815.

LIN, M., AND GOTTSCHALK, S. 1998. Collision detection between geometric models: A survey. In *Proc. of IMA Conf. on Math. of Surfaces*, 37–56.

MILENKOVIC, V. J., AND SCHMIDL, H. 2001. Optimization-based animation. In *Proc. SIGGRAPH 2001*, 37–46.

MILENKOVIC, V. J. 1996. Position-based physics: simulating the motion of many highly interacting spheres and polyhedra. In *Proc. SIGGRAPH 96*, 129–136.

MIRTICH, B., AND CANNY, J. 1995. Impulse-based dynamic simulation. In *Alg. Found. of Robotics*, A. K. Peters, Boston, MA, K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, Eds., 407–418.

MIRTICH, B., AND CANNY, J. 1995. Impulse-based simulation of rigid bodies. In *Proc. of 1995 Symp. on Int. 3D Graph.*, 181–188, 217.

MIRTICH, B. 1996. Fast and accurate computation of polyhedral mass properties. *J. Graph. Tools 1*, 2, 31–50.

MIRTICH, B. 2000. Timewarp rigid body simulation. In *Proc. SIGGRAPH 2000*, 193–200.

MOORE, M., AND WILHELMS, J. 1988. Collision detection and response for computer animation. *Comput. Graph. (Proc. SIGGRAPH 88) 22*, 4, 289–298.

MUSETH, K., BREEN, D., WHITAKER, R., AND BARR, A. 2002. Level set surface editing operators. *ACM Trans. Graph. (SIGGRAPH Proc.) 21*, 3, 330–338.

OSHER, S., AND FEDKIW, R. 2002. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag. New York, NY.

PANDOLFI, A., KANE, C., MARSDEN, J., AND ORTIZ, M. 2002. Time-discretized variational formulation of non-smooth frictional contact. *Int. J. Num. Meth. in Eng. 53*, 1801–1829.

PENTLAND, A., AND WILLIAMS, J. 1989. Good vibrations: modal dynamics for graphics and animation. *Comput. Graph. (Proc. SIGGRAPH 89) 23*, 3, 215–222.

PONAMGI, M. K., MANOCHA, D., AND LIN, M. C. 1995. Incremental algorithms for collision detection between solid models. In *Proc. ACM Symp. Solid Model. and Appl.*, 293–304.

POPOVIĆ, J., SEITZ, S. M., ERDMANN, M., POPOVIĆ, Z., AND WITKIN, A. 2000. Interactive manipulation of rigid body simulations. In *Proc. SIGGRAPH 2000*, 209–217.

REDON, S., KHEDDAR, A., AND COQUILLART, S. 2002. Fast continuous collision detection between rigid bodies. *CGForum 21*, 3, 279–288.

SAUER, J., AND SCHÖMER, E. 1998. A constraint-based approach to rigid body dynamics for virtual reality applications. In *Proc. ACM Symp. on Virt. Reality Soft. and Tech.*, 153–162.

SCHMIDL, H. 2002. *Optimization-based animation*. PhD thesis, University of Miami.

SCHROEDER, W. J., LORENSEN, W. E., AND LINTHICUM, S. 1994. Implicit modeling of swept surfaces and volumes. In *Proc. of Vis.*, IEEE Computer Society Press, 40–55.

SCLAROFF, S., AND PENTLAND, A. 1991. Generalized implicit functions for computer graphics. *Comput. Graph. (Proc. SIGGRAPH 91) 25*, 4, 247–250.

SETHIAN, J. 1996. A fast marching level set method for monotonically advancing fronts. *Proc. Natl. Acad. Sci. 93*, 1591–1595.

SIMS, K. 1994. Evolving virtual creatures. In *Proc. SIGGRAPH 94*, 15–22.

STEWART, D. E., AND TRINKLE, J. C. 2000. An implicit time-stepping scheme for rigid body dynamics with coulomb friction. In *IEEE Int. Conf. on Robotics and Automation*, 162–169.

STEWART, D. E. 2000. Rigid-body dynamics with friction and impact. *SIAM Review 42*, 1, 3–39.

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

TSITSIKLIS, J. 1995. Efficient algorithms for globally optimal trajectories. *IEEE Trans. on Automatic Control 40*, 1528–1538.

WEBB, R., AND GIGANTE, M. 1992. Using dynamic bounding volume hierarchies to improve efficiency of rigid body simulations. In *Comm. with Virt. Worlds*, CGI Proc. 1992, 825–841.

WESTERMANN, R., KOBBELT, L., AND ERTL, T. 1999. Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. *The Vis. Comput. 15*, 2, 100–111.