Penalty Force for Coupling Materials with Coulomb Friction

Ounan Ding and Craig Schroeder

Abstract—We propose a novel penalty force to enforce contacts with accurate Coulomb friction. The force is compatible with fully-implicit time integration and the use of optimization-based integration. The contact force is quite general. In addition to processing collisions between deformable objects, the force can be used to couple rigid bodies to deformable objects or the material point method. The force naturally leads to stable stacking without drift over time, even when solvers are not run to convergence. The force leads to an asymmetrical system, and we provide a practical solution for handling these.

Index Terms—Computer Graphics, Animation, Nonlinear programming

1 Introduction

Collisions between objects are a very important visual phenomenon, and the problem of resolving them has been explored extensively within computer graphics. Methods for resolving collisions can be classified broadly into three categories: Constraint-based approaches, penalty methods, and impulse-based methods.

Constraint-based approaches formulate contacts as algebraic constraints. These methods come in a wide variety, based on how the constraints are formulated, how friction is treated, the type of numerical problem that results, and how that problem is solved. The simplest formulations ignore friction during contact resolution, treating collisions as simple constraints [1], [2] and applying friction as a post-process. These methods are attractive because they yield problems that can be solved efficiently. Coevoet et al. [3] used a frictionless formulation with linearized finite element forces to achieve real time simulations of soft robots. Including friction in the formulation complicates the formulation significantly, resulting in linear or nonlinear complementarity problems (LCP or NCP) [4], [5], [6], [7]. This approach is particularly popular for rigid bodies, which are well-suited to this formulation due to the long-range effects of contacts and collisions in systems and the difficulty of resolving them simultaneously, but formulations with deformable objects are also possible [8]. Methods based on LCP or NCP formulations tend to scale poorly with the number of constraints n, scaling as $O(n^2)$ or worse [9]. The difficulty arises because the problem of resolving contact constraints is combinatorial in nature. Indeed, the general problem of computing contact forces to satisfy normal and frictional constraints is NP-hard [4]. The problem may also have no solution, necessitating the use of impulses to guarantee a solution. When collisions are decoupled from internal forces for deformable objects, implicit forces can be computed independently per object and in parallel, which can help offset the scaling with respect to the number of constraints [10].

Penalty formulations instead treat contacts as elastic forces, which are integrated alongside other forces. Due to their simplicity, penalty formulations are quite old [1], [11], [12], [13]. These methods are attractive because the computational cost of applying penalty forces scales linearly with the number of constraints, making them a popular choice for haptics [14]. A major limitation is the need to tune penalty stiffnesses, which represent a trade-off between performance and accuracy. Penalty methods normally allow some degree of collision to exist, though a stiff barrier potential is sometimes used to prevent this at additional computational cost. In this paper, we will use a penalty formulation because of its flexibility and because it allows us to naturally couple contact forces with elastic forces.

Impulse-based formulations represent a third approach to resolving contact. These methods work by resolving contacts one by one, iterating until some convergence criterion is achieved. This strategy is able to include friction and coefficient of restitution in a simple way. Due to its simplicity, this approach has been widely used, such as for cloth [15], [16] or rigid bodies [17]. Due to the iterative nature, these methods tend to be biased to some degree on the order in which collisions are processed, though many strategies exist to limit or avoid this. Impulse-based contacts are generally processed separately from elastic forces.

1.1 Penalty methods

Penalty forces compute contact forces as functions of some measure of penetration depth or close approach. [18] propose a penalty force model for contact between a robot and a ground based on linear damper springs. [19] proposed a penalty formulation based on a nonlinear damper spring, which they construct to avoid force discontinuities when contacts are created or lost. [20] adapt this nonlinear damping spring to the general contact problem between polygonal objects. They propose a penalty force model for contacts that uses damper springs connected to attachment points to enforce frictional contact. Their attachment points move when dynamic friction is applied, a strategy we also employ. Their penalty force is designed with the advantages and limitations of explicit time integration in mind. The characteristics of explicit time integration dictate many of the

• *O. Ding and C. Schroeder are with the University of California Riverside. E-mail:{oding001,craig}@ucr.edu*

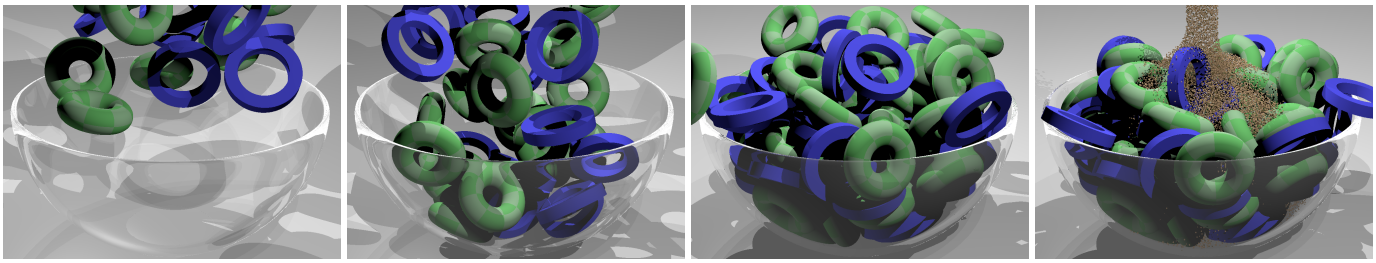


Fig. 1. Rigid bodies are dropped into a bowl, and sand is poured on top.

core design decisions, including a careful treatment to avoid applying too much dynamic friction, which could nonphysically flip the tangential sliding direction. The method of [21] adapts the penalty force model of [20] to the problem of robustly simulating problems involving large, time-varying contact areas. They use semi-implicit integration, including contact forces in their implicit integration but applying frictional forces explicitly.

Implicit friction The explicit friction formulations used by these methods are quite simple. It would seem tempting to include these models in the nonlinear force computations, as indeed [21] suggests as a possibility. In practice, this is a more difficult proposition than it may seem. In the case of [21], their implicit formulation solves a linearization of the problem (one step of Newton), followed by explicit (and very nonlinear) friction. The non-smooth nature of the static-dynamic transition is incompatible with the linearized nature of such a solver; the choice of static vs dynamic would need to be made *before* finding out whether the friction cone would actually be violated, which could result in non-physical strong sticking, tangential velocity reversal, or energy gain. This problem could be overcome by using multiple Newton iterations, which would give the solver a chance to correctly decide between static and dynamic friction. This solution in turn leads to Newton convergence problems (friction is non-smooth), line searches (or other stabilization techniques), and the need to gracefully handle configurations far from the current one (these occur occasionally when the trial Newton step is poor, during line searches, etc.). Solving these impediments to implicit friction is a major contribution of this paper.

An alternative strategy is to formulate the penalty force using an estimate of the volume of contact [9], [14]. This formulation emerged as an alternative to methods that computed penalty forces only for the deepest penetrating point, since this leads to significant artifacts such a chattering. This problem is largely avoided by applying forces to many or all penetrating particles, though a degree of resolution-dependence remains. Due to the complications involved with harmonizing attachment points and contact volume, we pursue a depth-based formulation and apply forces to all penetrating particles.

The computation of depth for deformable objects is nontrivial and has received significant attention [22]. Of particular relevance is [23], which used continuous collision detection to compute colliding point-face and edge-edge pairs. Their algorithm is explicit and includes Coulomb friction. Once computed, forces are computed along a chosen normal direction between pairs and persist until the contact is resolved.

1.2 Coupling

Part of our motivation for pursuing a penalty force formulation is its ability to couple different types of solid materials. In this paper, we consider coupling of rigid bodies to deformable bodies or to the material point method (MPM).

Many methods have been considered for coupling rigid bodies to deformable bodies [24], [25], [26], [27] to varying degrees. For example, [28], [29] allow rigid bodies and deformable bodies to exchanges forces and be embedded but do not consider contacts between them. The method of [30] treats collisions using a combination of impulse-based methods but also includes some contact forces in their implicit solves. More recent method [29], [31] achieve interactive or real-time rates through model simplifications.

Coupling between rigid bodies and MPM has received much less attention. MPM was original developed by Sulsky [32], [33] as an extension of hybrid PIC/FLIP fluid schemes to viscoelastic materials. Since its introduction to graphics by [34] as a way to simulate snow, MPM has become an increasingly popular simulation choice for complex materials, including sand [35], [36]. The method of [36] was the first to couple MPM with rigid bodies. Their method couples MPM-based sand to rigid bodies by unifying the internal frictional contact forces from sand with frictional contact forces with rigid bodies. (Sand was coupled to rigid bodies originally in [37], but they did not simulate sand with MPM.) Because the coupling of [36] relies on the sand constitutive model, it is not a general coupling method and cannot be applied to other MPM-based materials.

1.3 Tight coupling

In addition to coupling between materials, we are interested in tight coupling between elastic and contact forces. For

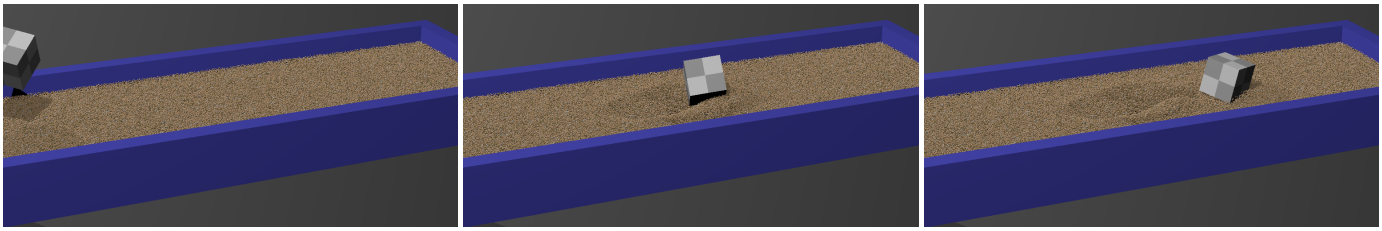


Fig. 2. In this simulation, we throw a cube over a sand box. The cube tumbles through the sand and comes to rest.

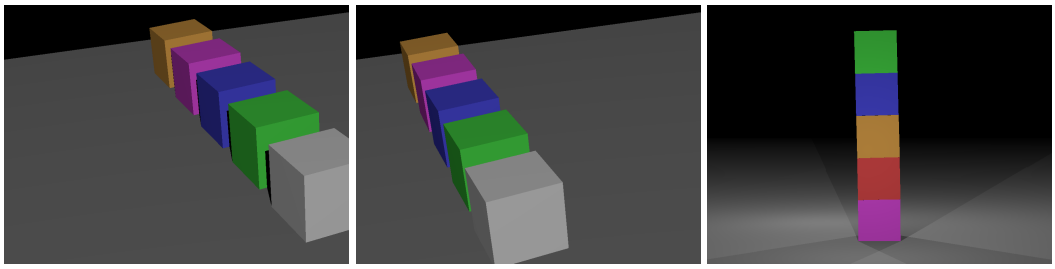


Fig. 3. (Left and middle) Analytic friction tests showing MPM/level set (orange), MPM/rigid body (magenta), rigid/rigid (blue), and rigid/deformable (green) contacts sliding to the left while obeying the analytic solution (silver). (Right) A stack of mixed objects remains standing indefinitely (rigid: magenta, blue, green; deformable: red, orange).

some phenomena, such as stacking objects in a pile (See Fig. 6, where a rigid body is stacked on three deformable (Lagrangian or MPM) objects), the coupling between elastic forces, normal contact forces, and friction plays a critical role in the resulting dynamics and long-term stack stability. If the bottom objects move out, then the top object will move down; this downward motion is progress towards collapse which is physically irreversible due to conservation of energy. A method that treated friction as a post-process would allow the pile to fall slightly (the correct behavior in the absence of friction) converting the gravitational potential into kinetic energy, which is then dissipated as friction during the post-process. An iterative method (impulse-based, partitioned) can leave small residual errors, which will accumulate over time until collapse. Even many analytic methods for systems with only rigid bodies struggle with long-term pile stability [38]. A similar problem occurs with wedging (see Fig. 8).

1.4 Our contribution

We propose a novel penalty force with the following properties.

- The force accurately enforces Coulomb friction, including both static and dynamic friction as well as transitions between them. Where this cannot be done, the force falls back gracefully.
- The force is compatible with use with fully-implicit time integration, including the special requirements of line-search-based methods (unlike existing penalty methods).
- The force can be used to couple rigid bodies to deformable objects or to the material point method.
- The force naturally produces stable stacking with zero drift over time, even if implicit integration is solved only approximately (unlike impulse-based methods or most existing constraint-based methods).
- We demonstrate a practical implicit stabilized Newton-based solver capable of solving the system of equations that results from backward Euler. The equations are especially challenging due to the inclusion of friction and are non-smooth with asymmetrical derivatives. The solver is also capable of stabilizing elastoplastic simulations, such as those that arise from the Drucker-Prager constitutive model for sand.

2 Penalty Force

We propose a penalty friction force to model collisions and contacts between *particles* and *objects*. The flexibility of the proposed force comes from the variety of representations that can be treated as particles or objects.

Particles. The particles can be almost anything Lagrangian. In our examples, we use (1) degrees of freedom from a deformable object simulation, (2) particles from a material point method (MPM) simulation, and (3) vertices from the surface mesh of a rigid body. Note that in cases (2) and (3), the *particles* are distinct from the actual degrees of freedom on which the integration of forces will occur (grid nodes for MPM, velocity and angular velocity for rigid bodies). It is sufficient for the velocities (and positions) of these *particles* to be computed from the degrees of freedom. Following the principle of virtual work, we use the transpose of this mapping to apply forces.

Objects. Suitable choices for the objects are a bit more restrictive, since the formulation of the penalty force must be tailored to properties of the object. We formulate versions of the penalty force for level sets and triangular surface meshes. Using these, we demonstrate collisions against fixed objects and boundaries, rigid bodies, and deformable objects. We use these options to demonstrate a range of capabilities, including deformable object self-collisions, rigid-rigid collisions and contacts, rigid-deformable coupling, rigid-MPM coupling, and handling of collisions with fixed objects for all simulation types.

Spring force. We formulate our force as a spring force connecting the *particle* (at location $\mathbf{Z}$) with an *attachment point* on the object (at location $\mathbf{X}$). The attachment point is able to slide along the surface of the object to a potentially new location $\mathbf{Y}$, a process we will refer to as *relaxation*. For the force itself we simply use a zero-length spring, where the force applied to the particle $\mathbf{Z}$ is $\mathbf{f} = k(\mathbf{Y} - \mathbf{Z})$. If the object is dynamic, an equal and opposite force is applied at the relaxed attachment location $\mathbf{Y}$. The constant k is the stiffness of the penalty force, which we choose as a compromise between the depth of penetration and the amount of stiffness in the resulting system.

Enforcing friction. The attachment point may be thought of as the place that the particle *should* be, and the spring applies a force on the particle to pull it there. The attachment point behaves as a massless point that collides with the object by exchanging normal and tangential forces subject to Coulomb friction. This massless point is not solved for as a degree of freedom; rather, its position is computed by *relaxing* from its original location $\mathbf{X}$ to a new location $\mathbf{Y}$ which satisfies the Coulomb friction cone. This relaxation step is a nonlinear projection operation (at least approximately), which we will denote as $\mathbf{Y} = \mathbf{P}(\mathbf{X}, \mathbf{Z})$. Eliminating $\mathbf{Y}$, we get the penalty force $\mathbf{f} = k(\mathbf{P}(\mathbf{X}, \mathbf{Z}) - \mathbf{Z})$,

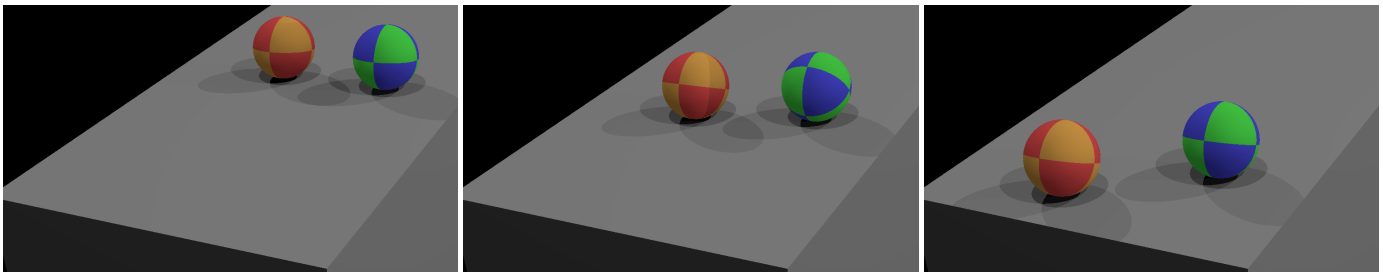


Fig. 4. Our frictional contact force handles rolling friction automatically. The blue/green sphere rolls due to friction, while the red/orange sphere has a very low friction coefficient and slides. Note that the orange sphere falls down the incline faster, as would be expected.

where $\mathbf{X}$ is fixed and k is a constant. Note that the projection operator $\mathbf{P}$ will depend on the degrees of freedom of the object, which may itself be dynamic. At the end of the time step, we update the attachment point ($\mathbf{X}^n \rightarrow \mathbf{X}^{n+1}$) using $\mathbf{X}^{n+1} = \mathbf{Y} = \mathbf{P}(\mathbf{X}^n, \mathbf{Z}^{n+1})$. What remains is to formulate this projection, which takes different forms for different types of objects. We provide pseudocode in a separate technical document for the force and relaxation routines, as well as their derivatives. To simplify notation, we will omit superscripts on $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$. The original attachment location $\mathbf{X}$ is fixed, the particle position $\mathbf{Z}$ is being computed by a newton solve, and the relaxed attachment location $\mathbf{Y}$ is computed when needed from $\mathbf{X}$ and $\mathbf{Z}$.

Representation of attachments. In the case of moving objects, the attachment point must be fixed to the object in a meaningful way. For rigid bodies, we store the attachment point in the object space of the rigid body. For deformable objects, we represent the attachment by its barycentric coordinates in the triangle that the particle is colliding with. In this way, attachment points naturally move with the object; when static friction is being applied, the representations of attachment points are untouched.

Mechanism of stable stacking. The means by which stable stacking is achieved is worth emphasizing. In constraint-based or impulse-based formulations, small movements of colliding points across a surface (due to, among other things, convergence error) may lead to long-term drift of the colliding point over the surface. This may in turn cause piles or stacks to collapse. Methods that use penalty forces with attachment points behave quite differently. The colliding particles are still able to move around (in both normal and tangential directions), but these particles are tethered to an attachment point. As long as the attachment point is prevented from moving relative to the object (which we accomplish by storing the attachment *in the reference space of the object*), no long-term progress towards collapse is possible. The particle can only move around in a small region near its attachment point.

2.1 Relaxation - properties

Let $\mathbf{Y} = \mathbf{P}(\mathbf{X}, \mathbf{Z})$ be the relaxed (projected) attachment point and $\mathbf{f} = k(\mathbf{Y} - \mathbf{Z})$ the resulting force. If $\mathbf{n} = \mathbf{n}(\mathbf{Y})$ is the local normal direction of the object at the projected attachment location, then our point should satisfy the following properties.

- $\mathbf{Y}$ is on the surface on the object.
- The projection is idempotent: $\mathbf{Y} = \mathbf{P}(\mathbf{Y}, \mathbf{Z})$.
- The friction cone is satisfied: $\|\mathbf{f} - (\mathbf{f} \cdot \mathbf{n})\mathbf{n}\| \leq \mu \mathbf{f} \cdot \mathbf{n}$.
- The attachment point should not be relaxed further along the surface than necessary to satisfy the friction cone.

Note that the last requirement is a slightly weakened form of the usual complementary condition, which is necessary to accommodate a non-smooth object surface. Since the force is linear in positions, the friction constraint is equivalent to the more convenient geometrical constraint

$$\|(\mathbf{Y} - \mathbf{Z}) - ((\mathbf{Y} - \mathbf{Z}) \cdot \mathbf{n})\mathbf{n}\| \leq \mu(\mathbf{Y} - \mathbf{Z}) \cdot \mathbf{n}. \quad (1)$$

The choice $\mathbf{n} = \mathbf{n}(\mathbf{Y})$ was not the only option available for computing a normal direction; Fig. 5 motivates the appropriateness of this choice.

2.1.1 *Relaxation with a plane*

We begin with the simple case where the object is a plane (See Fig. 5). If (1) is satisfied with $\mathbf{Y} = \mathbf{X}$, then the attachment experiences static friction and does not move. Otherwise, the attachment $\mathbf{X}$ should be moved along the

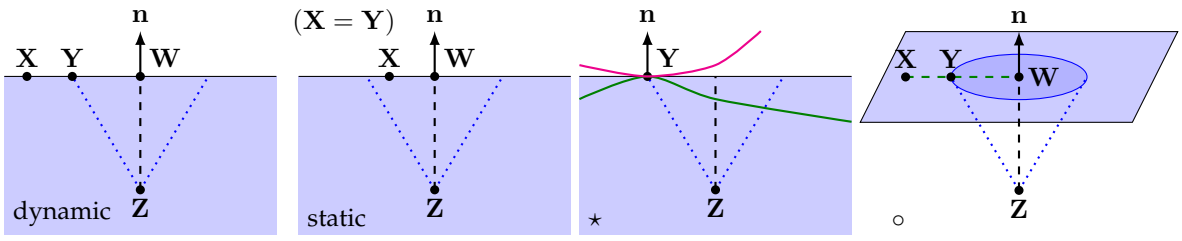


Fig. 5. A particle $\mathbf{Z}$ as pulled to the surface by an attachment point $\mathbf{X}$. In dynamic friction, the attachment point $\mathbf{X}$ is outside the friction cone (dotted) and relaxes along the surface to the friction cone $\mathbf{Y}$, resulting in a penalty force that satisfies Coulomb friction. In static friction, the attachment point $\mathbf{X}$ is already inside the friction cone and does not move ($\mathbf{X} = \mathbf{Y}$). In the non-planar case ($\star$), there is not a single constant normal direction. Intuitively, the attachment at $\mathbf{Y}$ should lie on the friction cone for the planar surface as well as the green and red curved surfaces. That is, friction is a local property that depends on the surface at only one location. Viewed in 3D ($\circ$), the attachment $\mathbf{X}$ is relaxed towards the closest point on the plane $\mathbf{W}$ until it reaches the cone at $\mathbf{Y}$, since the projection of a force from $\mathbf{X}$ to $\mathbf{Z}$ onto the surface points in this direction.

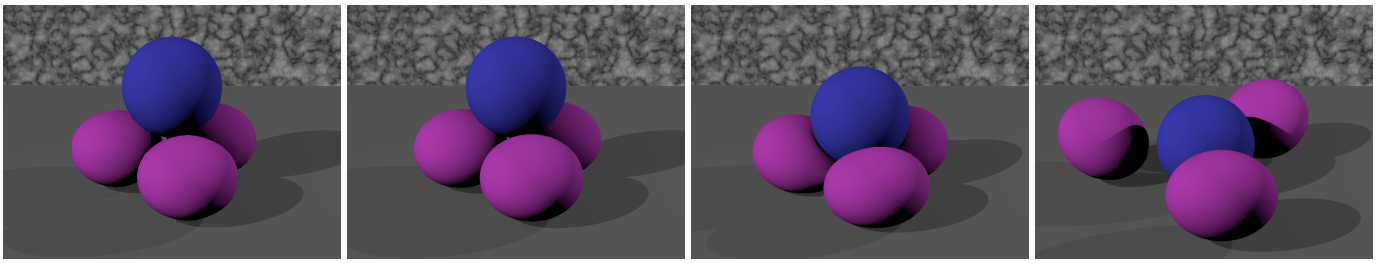


Fig. 6. Our method maintains stable stacks in configurations that rely on force balance between elastic forces, contact, and friction. Stable stacking is shown with rigid/deformable (left) and MPM/rigid (left middle). If friction is reduced, the stack collapses as expected (right two).

surface by the penalty force $\mathbf{f}$ until (1) is satisfied. Let $\mathbf{W} = \mathbf{Z} - \phi(\mathbf{Z})\mathbf{n}$ be the closest point on the plane to $\mathbf{Z}$, where ϕ is the level set function for the planar object. This force pulls $\mathbf{X}$ along the surface towards $\mathbf{W}$ to the point $\mathbf{Y} = \mathbf{W} + s(\mathbf{X} - \mathbf{W})$, where s is the largest scalar $0 \leq s \leq 1$ satisfying (1). Solving this equation yields

$$s = \mu \frac{\|\mathbf{Z} - \mathbf{W}\|}{\|\mathbf{X} - \mathbf{W}\|}. \tag{2}$$

2.1.2 Relaxation with level set

If we are colliding a particle with a level set, and that level set represents a plane, then we have a simple algorithm to compute $\mathbf{Y}$ directly. More generally, the level set will not be flat. Relaxing the attachment based on the local force would amount to solving an ODE, which is unnecessarily complicated. Instead, we relax the attachment point towards the closest point $\mathbf{W} = \mathbf{Z} - \phi(\mathbf{Z})\mathbf{n}(\mathbf{Z})$ on the surface of the object as we did in the plane case. Unlike the plane case, a point $\mathbf{K} = \mathbf{W} + s(\mathbf{X} - \mathbf{W})$ along the segment connecting two points $\mathbf{X}$ and $\mathbf{W}$ on the surface need not be on the surface. We project this point to the surface to compute the desired relaxed point $\mathbf{Y} = \mathbf{K} - \phi(\mathbf{K})\mathbf{n}(\mathbf{K})$.

The attachment point $\mathbf{Y}(s)$ is a nonlinear function of a single scalar $0 \leq s \leq 1$. Let

$$g(s) = ((\mathbf{Y} - \mathbf{Z}) \cdot \mathbf{n}(\mathbf{K}))^2 - \bar{\mu}\|\mathbf{Y} - \mathbf{Z}\|^2 \qquad \bar{\mu} = \frac{1}{\mu^2 + 1}.$$

Then $g(s) \geq 0$ is equivalent to (1). If $g(1) \geq 0$ then $\mathbf{Y}(1) = \mathbf{K} = \mathbf{X}$ satisfies (1); the attachment experiences static friction and does not move. Otherwise, $g(1) < 0$. Observe that $\mathbf{Y}(0) = \mathbf{W}$ and $\mathbf{n}(\mathbf{W}) = \mathbf{n}(\mathbf{Z})$ so that $g(0) = (1 - \bar{\mu})\phi(\mathbf{Z})^2 \geq 0$. If $g(s)$ is continuous, a suitable s must exist, and a method such as bisection may be used to compute it. In practice, $g(s)$ need not be continuous (e.g., at sonic points of the level set), and bisection may terminate at a discontinuity. In this case, the friction cone is only weakly satisfied. While differentiating this procedure is manageable, doing so robustly is quite difficult.

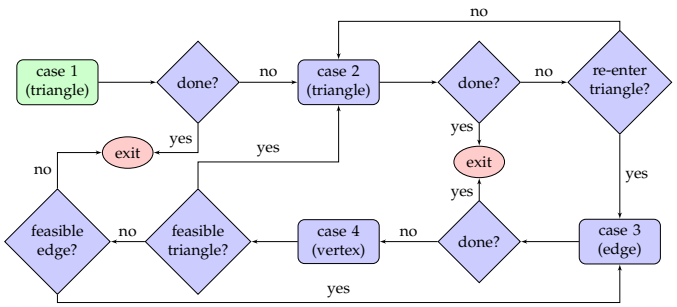


Fig. 7. This figure shows the case transitions for the mesh-based relaxation algorithm. The algorithm begins at the green node and ends when reaching a red node.

Instead, we observe that we are typically only moving points a small distance during a time step. Locally, the level set should look like a plane. Motivated by this, we simply compute s using (2) rather than by bisection. This approximation gives up the projection property and means that Coulomb friction is only approximately satisfied, but it seems to be a good comprise in practice.

Initial attachment location. When a collision first occurs, an attachment location $\mathbf{X}$ must be chosen. Ideally, one would chose $\mathbf{X}$ to be the location on the surface of the level set where the particle crossed it. To avoid complications for rigid bodies, we simply use $\mathbf{X} = \mathbf{Z} - \phi(\mathbf{Z})\mathbf{n}(\mathbf{Z})$; this contrasts with the CCD case, where a good initial attachment location is readily available. While this is less accurate than using the point of entry, this only introduces an error in the time step in which the collision is first encountered. In subsequent time steps, the attachment location $\mathbf{X}$ will not move relative to the object (static friction) or will relax across the surface (dynamic friction) according to Coulomb friction.

Note that no friction will be applied when the particle is at the location where the collision was first detected (or more generally whenever the particle happens to be below the attachment point). The lack of force under these conditions does not mean the particle is not being tethered to this attachment location. The particle will feel strong frictional forces if it ever attempts to leave the vicinity of the attachment point. This is not particularly surprising; a stationary box on a level surface also experiences zero frictional force. The box will only experience non-zero frictional forces if one tries to move it.

Limitations of the level set formulation. The primary limitation of the level set formulation is the discontinuities caused by sonic points. The formulation works well when the surface has low curvature. This keeps sonic points away from the surface and also makes the approximate computation of s more accurate.

2.1.3 Relaxation with surface mesh

When colliding against a level set, we can immediately determine whether a collision is occurring and project to the closest point on the surface. When we are colliding with a triangulated surface mesh, these operations are more expensive.

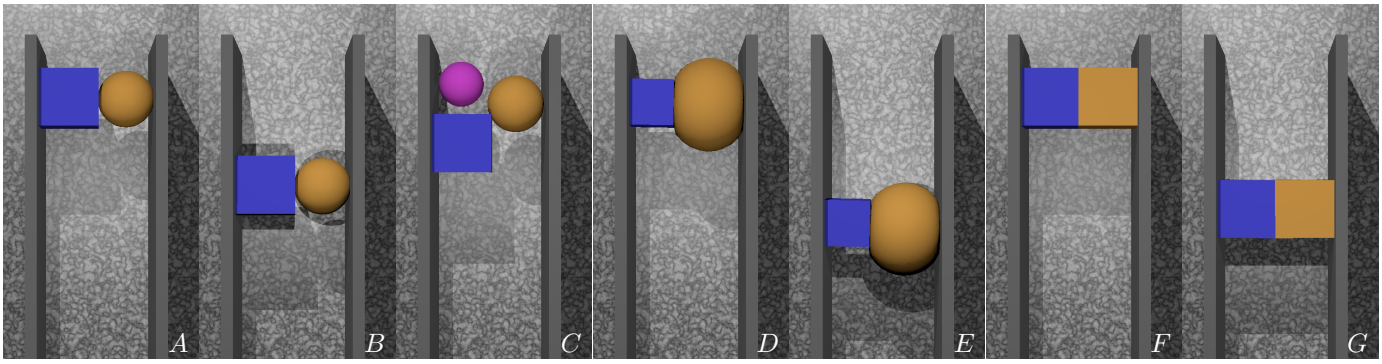


Fig. 8. Our forces pass wedge tests, where objects are prevented from falling by virtue of being wedged between two parallel plates with friction. The first three figures shows a wedged deformable/rigid configuration (A); if friction is reduced, the objects slide (B). The wedged objects may also be dislodged by other collisions (C). We also demonstrate wedging with MPM/rigid (D and E) and rigid/rigid (F and G), first in a wedged configuration and then sliding when friction is reduced. The blue blocks and magenta spheres are rigid.

Detection. We use continuous collision detection (CCD) to detect when a particle has penetrated a triangulated surface. This conveniently gives us the barycentric coordinates for the point of entry as well, which we use to set the initial attachment $\mathbf{X}$. CCD can normally only be employed to detect new collisions, and CCD algorithms expend great effort to ensure that a collision-free state is maintained. This is because CCD will not detect a colliding particle if it was already colliding. In our case, a collision-free state is not maintained; since our force is a penalty force, forces are only applied if a small amount of penetration is retained. This is not a problem for us, since CCD will correctly flag new collisions without a collision-free state. We do not need CCD to tell us about existing collisions, since we keep a record of these interactions anyway (we must store attachment locations for them). Depending on the implementation, CCD may also detect when a colliding particle exits the surface; these can be easily detected by checking the triangle normal and may be safely ignored.

Intuition for relaxation. As intuition for the relaxation process we formulate, imagine that the attachment point is a block sitting on the surface and connected to the colliding point $\mathbf{Z}$ with an actual spring. The block experiences Coulomb friction against the surface mesh. Ignoring inertial effects, the attachment point should slide along the surface of the triangle mesh along the direction it is being pulled by the spring force. This process terminates when the friction cone is satisfied. At this point, we check to see if $\mathbf{Z}$ is pulling $\mathbf{Y}$ into the surface or away from the surface. If the latter, we flag the attachment as inactive and do not apply forces to it.

Convexity consideration. In the case of non-convex objects, one has to make a choice about when the separation test should be performed, since an attachment may be pulled away from the object during relaxation, even though the colliding point is in fact inside the object. One option would be for the attachment to be able to separate from the surface during relaxation (and possibly re-collide with it). Another option is to delay the separation test until relaxation has completed. In practice, we found the latter option to be simpler to implement, more efficient, and more effective. See the technical document for details of the separation tests.

Finite state machine. We implement the resulting relaxation algorithm as a finite state machine (FSM) with four states: (1) the attachment point is in the interior of a triangle, (2) the attachment point is on an edge of a triangle, (3) attachment point is on an edge and moves along it, and (4) the attachment point is at a vertex and tries to leave it via a neighboring primitive. At each step, we start with an initial attachment $\mathbf{Y}^{(k)}$ on the current primitive and compute a new attachment $\mathbf{Y}^{(k+1)}$ location. The algorithm starts in state (1) with $\mathbf{Y}^{(0)} = \mathbf{X}$. When the algorithm terminates at step n, we set $\mathbf{Y} = \mathbf{Y}^{(n)}$. These states are illustrated in Fig. 9 and Fig. 11 and described in detail below. The transitions are illustrated in Fig. 7.

Triangle cases. Cases (1) and (2) are nearly identical. In both cases, a relaxation $\mathbf{Y}^{(k)} \to \hat{\mathbf{X}}$ is performed along the plane of the triangle in the same way as for a planar level set. If the computed attachment point $\hat{\mathbf{Y}}$ lies inside the triangle, then the algorithm terminates with $\mathbf{Y} = \mathbf{Y}^{(k+1)} = \hat{\mathbf{Y}}$. Otherwise, we draw a segment from $\mathbf{Y}^{(k)}$ to $\hat{\mathbf{Y}}$ and intersect it with the triangle's boundary. The intersection location is $\mathbf{Y}^{(k+1)}$. If we are in case (1) or the edge intersected is not the same one we entered on (so that $\mathbf{Y}^{(k)} \neq \mathbf{Y}^{(k+1)}$), then we

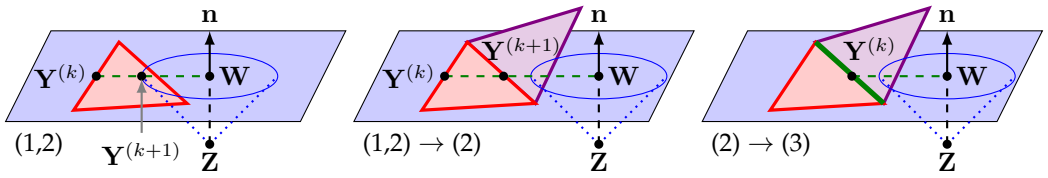


Fig. 9. In the triangle cases (1,2), the attachment starts at $\mathbf{Y}^{(k)}$, which is in the interior (case (1)) or on the edge (case (2)). The attachment is relaxed within the plane of the triangle; if the friction cone is reached within the triangle (left), the relaxation terminates. Otherwise, the attachment relaxes to the location where it leaves the triangle (middle); the algorithm enters case (2) with the neighboring violet triangle. In case (2), the attachment starts on an edge; if it is drawn outside the triangle, then the algorithm transitions to case (3) to relax along the green edge (right).

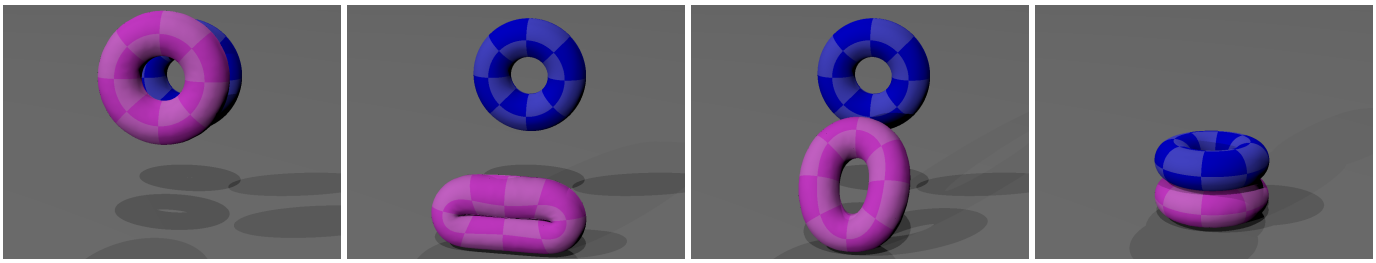


Fig. 10. Two tori (magenta is deformable, blue is rigid) are dropped to the ground, demonstrating the ability to handle non-convex objects, including self-collisions.

transition to state (2). Otherwise, we are stuck on an edge and unable to make progress along either adjacent triangle. We may still be able to make progress along the edge, and we transition to state (3). See Fig. 9.

Edge case. The edge case (3) is similar to case (1), except that the surface point $\mathbf{W}$ is computed as the closest point on the line containing the edge. If the computed attachment point $\hat{\mathbf{Y}}$ lies inside the segment, then the algorithm terminates with $\mathbf{Y} = \mathbf{Y}^{(k+1)} = \hat{\mathbf{Y}}$. Otherwise, we transition to case (4). See Fig. 11.

Vertex case. The vertex case (4) is quite different from the other cases, since the attachment point does not move. Rather, the goal of this state is merely to find a way to leave it. First, we check the triangles adjacent to the vertex. If the spring force would pull the attachment point into the triangle, then we transition to this triangle in state (2) with $\mathbf{Y}^{(k+1)} = \mathbf{Y}^{(k)}$. Note that it is not necessary to compute the actual attachment location in this triangle. If no progress can be made in a triangle, then we must check the adjacent edges. If the spring force would pull the attachment point along an edge, then we transition to this edge in state (3) with $\mathbf{Y}^{(k+1)} = \mathbf{Y}^{(k)}$. Otherwise, the attachment point would not be pulled from the vertex, and the algorithm terminates with $\mathbf{Y} = \mathbf{Y}^{(k)}$. See Fig. 11.

Termination considerations. Observe that if the attachment moves ($\mathbf{Y}^{(k)} \neq \mathbf{Y}^{(k+1)}$) then $\|\mathbf{Y}^{(k+1)} - \mathbf{Z}\| < \|\mathbf{Y}^{(k)} - \mathbf{Z}\|$. That is, the attachment point is being drawn closer to $\mathbf{Z}$. As as long as the attachment can be prevented from stalling in one spot, one can prove that primitives are visited at most a finite number of times (see the technical document for a proof). Assuming the triangle mesh is not degenerate, this can only occur when the attachment point is at a vertex. This can happen in practice due to round-off error. This can be easily checked, since the barycentric coordinates of the embedding are computed during case (2). We make two modifications to the algorithm to address this possibility: transition to case (4) if a barycentric weight is larger than $1 - 128\epsilon$ where ϵ is the floating point epsilon, and only transition away from case (4) if the test for progress is significantly larger than round-off error.

2.2 Collision detection

For level-set-based forces we detect collisions using the level set. We accelerate the search by rasterizing particles and level sets to regular sparse grids. For CCD-based forces, collisions are detected using CCD without a collision-free state. We use bounding box hierarchies to accelerate the search. In either case, we maintain hash tables of known pairs, which we use to avoid repeatedly registering the same pairs. In the case of CCD-based forces, we perform the hash table check on bounding box candidates before solving the cubic, since the hash table lookup is significantly cheaper. Once registered, each collision pair is retained along with its current attachment point until the collision becomes inactive. At the end of each time step, inactive collision pairs are pruned and attachment points for the remaining pairs are updated to their newly relaxed locations. Note that this postprocessing step is merely making permanent the decisions already made during the Newton solver. Collision pairs that are pruned at the end of the time step were inactive at the end of the Newton solve and thus were applying no force.

3 Stabilized Newton solver

We discretize our equations of motion using backward Euler, which leads to the nonlinear problem

$$\mathbf{M}\frac{\mathbf{v}^{n+1} - \mathbf{v}^n}{\Delta t} = \mathbf{f}(\mathbf{x}^{n+1}) \qquad \mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^{n+1},$$

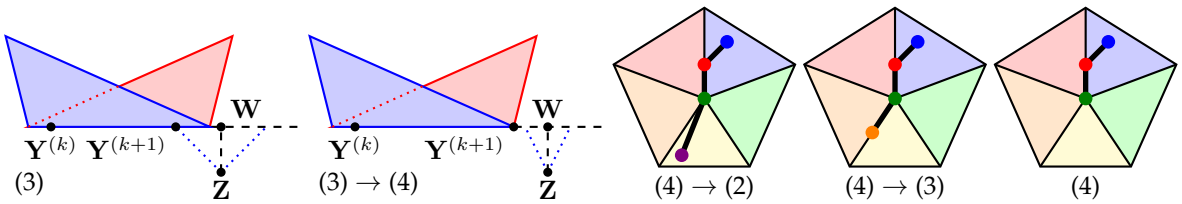


Fig. 11. In case (3), an attachment starts on an edge $\mathbf{Y}^{(k)}$ and relaxes towards the point $\mathbf{W}$ along the edge closest to the particle $\mathbf{Z}$. If the friction cone is reached along the edge, then the relaxation terminates (left). Otherwise, the attachment relaxes to a vertex and the algorithm resumes in case (4) by trying to leave the vertex (left middle). Case (4) handles relaxation through a vertex. This case is reached when an attachment (blue) gets stuck on an edge (red) and relaxes along it to a vertex (green). Case (4) transitions to case (2) if progress can be made (violet) in one of the neighboring triangles (middle). Otherwise case (4) transitions to case (3) if progress can be made (orange) along one of the neighboring edges (right middle). If no progress can be made, relaxation terminates (right).

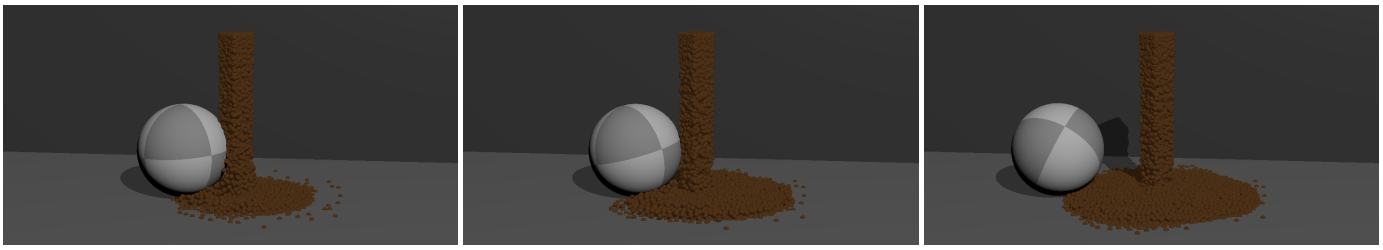


Fig. 12. A complex MPM material interacts with a rigid sphere.

where $\mathbf{M}$ is the lumped mass matrix and $\mathbf{f}(\mathbf{x})$ are our forces (gravity, internal, and collision forces). Using $\Delta\mathbf{v} = \mathbf{v}^{n+1} - \mathbf{v}^n$ as our degrees of freedom, we have $\mathbf{g}(\Delta\mathbf{v}) = \mathbf{M}\Delta\mathbf{v} - \Delta t\mathbf{f}(\mathbf{x}^n + \Delta t\mathbf{v}^n + \Delta t\Delta\mathbf{v}) = \mathbf{0}$. For simplicity, we refer to the solution vector as $\mathbf{z}$, where $\mathbf{z} = \Delta\mathbf{v}$. We refer to the accompanying technical document for details on the computation of $\mathbf{g}$ and its derivative.

Our contact formulation produces nonlinear systems that result in asymmetrical linear systems. We found that stabilizing our Newton solver greatly improved solver reliability. Stabilizing the solver had the added benefit of facilitating debugging, since it allows convergence problems to be tracked down to a definite source.

When using Newton's method to solve $\mathbf{g}(\mathbf{z}) = \mathbf{0}$, where $\mathbf{H} = \nabla\mathbf{g}$, we must repeatedly solve $\Delta\mathbf{z} = -\mathbf{H}^{-1}\mathbf{g}$ to obtain a series of corrections. For our forces, it is easy to see that $\mathbf{H}$ is not symmetric. This implies that $\mathbf{g} = \mathbf{0}$ does not correspond to minimizing anything, since otherwise $\mathbf{H}$ would be a Hessian of this objective and thus symmetric. This immediately rules out methods such as [2].

One known alternative is to minimize $E = \frac{1}{2}\|\mathbf{g}\|^2$ instead [39]. If $\mathbf{g} = \mathbf{0}$ has a solution, then clearly this will be a global minimum for E, at which point $E = 0$. One could optimize E directly, but this would involve computing the Hessian of E, which would in turn require us to compute second derivatives of forces.

Instead, we note that one can actually minimize E while continuing to use the line search direction $\Delta\mathbf{z} = -\mathbf{H}^{-1}\mathbf{g}$, as was done in [40]. This direction is usually very desirable; it is the direction we would use if we could not stabilize the solver. To be effective, we need this direction $\Delta\mathbf{z}$ to be a downhill direction for our new objective E. Consider that a step will be taken in some direction $\mathbf{u}$. The directional derivative of E in this direction is $\nabla E \cdot \mathbf{u} = \mathbf{g}^T\mathbf{H}\mathbf{u}$. Thus, a direction $\mathbf{u}$ is a downhill direction if $\mathbf{g}^T\mathbf{H}\mathbf{u} < 0$. We can immediately see that the Newton direction is *always* such a direction, since $\mathbf{g}^T\mathbf{H}\Delta\mathbf{z} = -\mathbf{g}^T\mathbf{H}\mathbf{H}^{-1}\mathbf{g} = -\mathbf{g}^T\mathbf{g} < 0$ (unless $\mathbf{g} = \mathbf{0}$, in which case we are done). This is not true of the standard objective, where this guarantee only holds where $\mathbf{H}$ is positive definite. In practice, computing the Newton direction exactly would be too expensive, and it need not actually exist (for example if $\mathbf{H}$ is singular). Instead, we test to see if it is downhill directly and choose a suitable fallback if it is not (see its accompanying technical document). Finally, we perform Wolfe condition line searches on E in the usual way. Our termination condition for Newton's method is that $\min(\|\mathbf{g}\|, \|\mathbf{H}^T\mathbf{g}\|) < \tau$, where τ is our Newton tolerance. Checking $\|\mathbf{g}\| < \tau$ ensures that we have an accurate solution to our problem (solve $\mathbf{g} = \mathbf{0}$). The test $\|\mathbf{H}^T\mathbf{g}\| < \tau$ ensures that we also have a good solution to the problem of minimizing E, the problem that the line search is trying to solve. Pseudocode for our solver is provided in the accompanying technical document.

While this method is only guaranteed to converge to a local optimum, in practice we have never observed convergence to anything other than an optimal solution. The sonic points of level sets cause force discontinuities, which in turn lead to discontinuities along the line search. This may cause convergence to fail for thin or sharp objects when using level sets. For this reason, we favor the CCD-based formulation for such objects.

4 Results

In our tests deformable objects in tetrahedral volume adopt fixed co-rotated constitutive model [41], with Young's modulus $E = 10^6$ and Poisson's ratio $\nu = 0.45$. Deformable objects in MPM material adopts the same model but with Young's modulus $E = 10^3$ and Poisson's ratio $\nu = 0.3$. Cloth uses mass-spring model [42], with linear stiffness $1000/(1+\sqrt{2})$ and bending stiffness $200/(1+\sqrt{2})$. The sand uses the Drucker-Prager model from [35] with Young's modulus $E = 35.37 \times 10^6$ and Poisson's ratio $\nu = 0.3$. We set tolerance of Newton's Method to be 1 by default, and use smaller tolerance in some tests to avoid visual artifacts. Detail configurations are shown in Table 1. The maximum number of Newton iterations is not limited (we set it to 1000). Based on our measurement the average number of iterations is 3.1 per time step.

Analytic friction tests. The heart of our MPM-rigid coupling method method is our penalty force for applying contact and friction. We show that our friction is accurate by comparing our force against the classical analytic solution for a point mass sliding along an inclined plane. In Fig. 3 we slide a block down an inclined plane with a variety of different object types (MPM, rigid bodies, and deformable objects), shown next to a proxy for the analytic solution. This demonstrates that we achieve accurate friction regardless of object representation. In Fig. 13, we show quantitative agreement with the analytic solution. This figure also demonstrates an important property of our method, that normal and frictional forces always obey the Coulomb friction cone. We have run this comparison with a tighter Newton tolerance to reduce deviations from the analytic solution caused by solver accuracy. In all of our inclined plane tests, the block is given an initial velocity $0.1m/s$ along the inclined plane. Depending on the friction used (see Table 1), the block continues sliding ($\mu = 0.1$) or comes to rest ($\mu = 0.125$). Our method naturally causes rigid bodies to roll (see Fig. 4).

Stacking. One of the effects that is difficult to handle correctly when elastic forces, friction, and contacts are not performed together is stacking. Our method produces stable

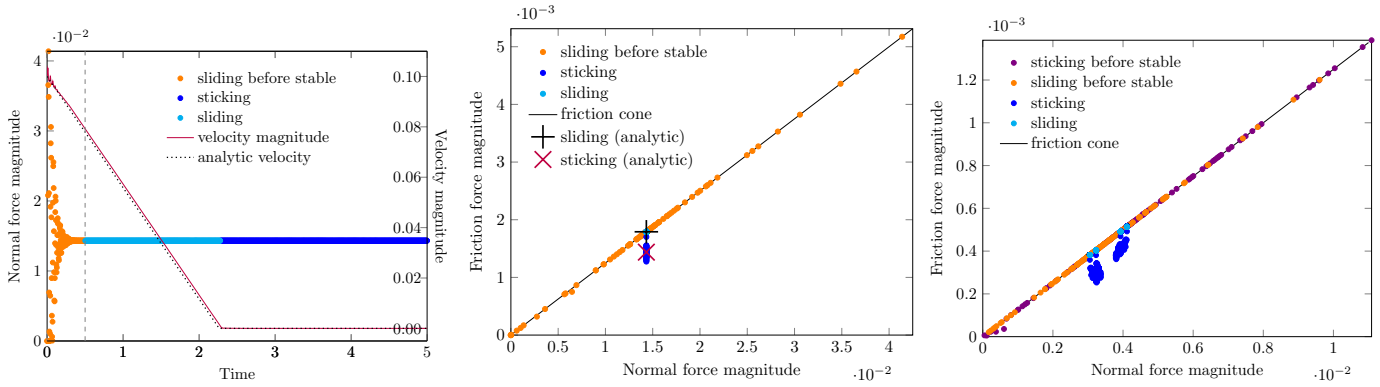


Fig. 13. In this figure, we compare our simulation of a deformable cube on an inclined plane (the green cube in Fig. 3, but with Newton tolerance 10^{-3}) with the analytic solution. (Left) In this plot, we show computed total normal contact force for the cube over time. At the beginning of the simulation the forces (•) are variable due to the bouncing of the cube as it falls from its initial stress-free configuration onto the inclined plane. After the cube has settled, the cube slides (•) until coming to rest (•). Superimposed are the analytic (·····) and computed (—) velocity magnitudes for the cube, showing close agreement between computed and analytic solution. Note that the abrupt change in the velocity magnitude slope corresponds with the switch from dynamic to static friction. (Middle) Here, we plot total normal and total tangential contact forces for the cube. During the settling phase (•), dynamic friction forces are variable but always on the friction cone (—). Afterwards, the computed dynamic (•) and static (•) friction forces closely match the analytic solution (+ and ×) and obey the Coulomb friction cone. (Right) Here, we plot total normal and total tangential contact forces per simulation vertex. During the settling phase (static •, dynamic •), friction closely tracks but never violates the friction cone (—). Once settled, the cube experiences dynamic (•) followed by static (•) friction. As expected, dynamic friction lies on the friction cone, and static friction lies below it. Note that the different vertices of the cube carry different amounts of the cube's mass, so the per-vertex contact force magnitudes cluster around different values. Our penalty contact force never violates the Coulomb friction cone.

stacking (see Fig. 3). Here we stack five blocks (a mixture of rigid and deformable objects), which stands stably and never falls (it remained standing for 100,000 frames, by which time it had come to rest). As long as contacts are in the sticking state, the attachments will never move. Because of this, the contacts will never drift, even over long periods of time. In Fig. 6, we form a pyramid by stacking a rigid ball on deformable or MPM objects. The stacks are stable and do not drift over time. This is a difficult test for many methods to pass. If elastic forces are evolved without friction, the objects at the base will slide out slightly, and the sphere will make some progress downward. This progress will not be corrected when friction is applied to the base objects. If the sphere is able to make downward progress or the base objects are able to make outward progress, the pyramid will eventually collapse. We tested the pyramid stack using the method of [30] and verified that it does indeed creep to collapse (see video).

Wedging tests. Another test that is difficult to pass without coupling between elastic forces, contact, and friction is the wedging test. In this test, two objects are squeezed by two fixed parallel vertical walls (see Fig. 8), and gravity is applied. Initially the objects overlap with the walls a little bit. This small penetration generates penalty forces. With adequate friction, the objects should be able to jam in place and never slide down the wall. As with the pyramid stack, the key to long term stability in this case is preventing the objects from making downward progress once they have become jammed. This jamming depends on the interaction of all three types of forces. We demonstrate jamming at higher friction and that sliding is recovered if friction is lowered.

Coupling with complex materials. Like [36], we are able to couple rigid bodies with sand. In Fig. 2, we throw a cube over a sand box. In Fig. 14, we roll a sphere down a sand pile. In Fig. 16, demonstrate that pouring sand on a sphere causes that sphere to roll, thereby demonstrating the frictional forces between them. In Fig. 12, demonstrate that we are able to couple to complex MPM materials other than sand. In Fig. 1, we demonstrate that we can scale to large numbers of objects.

General tests. Our method handles dynamic scenes, non-convex objects, and interactions between different materials. In Fig. 15, we demonstrate compatibility with cloth. In Fig. 10, we demonstrate that deformable-deformable and deformable-rigid contacts also work correctly with non-convex objects.

To judge the relative cost of our collision handling in comparison to other parts of the algorithm, we computed a breakdown of the runtime cost for three representative examples: MPM "sand_on_sphere" (Fig. 16), non-convex rigid/deformable "torus_dr" (Fig. 10), and rigid/deformable/cloth "cloth" (Fig. 15). For the MPM example, collision-related steps added only minor cost (4.0% for CCD, 0.6% for computing collision forces). The significant majority of the time was spend computing and applying internal sand forces (90.3%), with the remaining steps taking about 5% of the total time. The low cost of collisions in this example is due to three factors: the large number of particles involved in the sand, the relatively low number of particles close enough to collision objects to be involved in collision processing, and the cost of particle/grid transfers. For the non-MPM tests, continuous collision detection is a major part of the total cost (torus: 43.5%, cloth: 54.0%). In both cases, calculating collision forces, computing and applying collision force derivatives, and performing relaxation are minor (less than 2%). Computing and applying internal forces and other solver-related steps contribute most of the rest (torus: 55.4%, cloth: 44.0%).

To evaluate the convergence of the relaxation procedure, we consider the breakdown of states encountered during the non-convex example Fig. 10. The relaxation nearly always terminates at the starting state (approximately 99.77% of the time). Thus for performance purposes, the algorithm com-

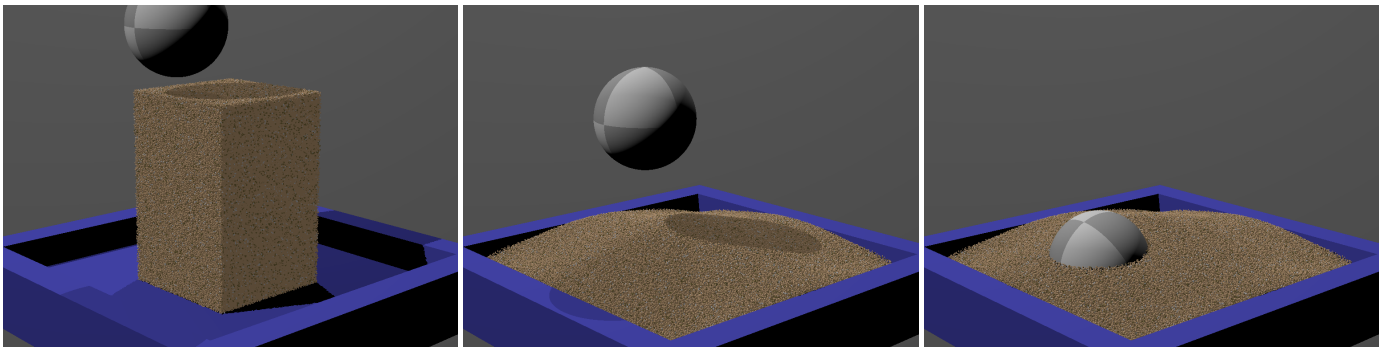


Fig. 14. Sand is dropped into a pile, and then a heavy sphere is dropped on time, penetrating into the sand.

pletes in one step, and the history that must be stored for derivatives is of constant size (see the technical document for details). For robustness, however, the remaining 0.23% cases must still be handled reliably. An single extra step (case $1 \rightarrow 2$) was taken in nearly all of the remaining cases (0.21%). The other case transitions that were observed in this example were (in order from most to least frequent): $1 \rightarrow 2 \rightarrow 3$, $1 \rightarrow 2 \rightarrow 2$, $1 \rightarrow 2 \rightarrow 2 \rightarrow 2$, and finally $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 2$ (which occurred three times in this simulation). No relaxation terminated in more than 5 steps for this example. This example illustrates that it is possible for the attachment to relax across multiple triangles in a single time step (up to four triangles in this example).

The constant k determines the stiffness of the penalty forces and is an important parameter in any penalty force model. Reducing k results in deep penetration, which produces observable artifacts and must be avoided. Increasing k produces linear systems with poorer conditioning, which makes solvers less accurate and slower to converge. Reduced accuracy from the linear solver is not a major concern in our case, since these errors will be corrected in the next Newton iteration. However, the slower convergence is a major concern, and careful tuning of parameters (both penalty stiffnesses and other solver parameters) can improve solver performance significantly. We observed that acceptable results can be obtained with only very crude parameter tuning. Indeed, most of our tunable parameters are simply powers of ten (See Table 1).

We run the test "torus_dr" with a range of coefficients of friction (0.1, 0.2, 0.5, 0.6, 1.0 and 10) and a range of stiffness (10, 10^2, 10^3 and 10^4). The method is able to accurately simulate dynamics at any coefficient of friction. The performance is not sensitive to the coefficient of friction. The simulations are stable and convergent in all cases, though convergence problems are observed for the highest stiffness ($k = 10^4$). Performance is strongly dependent on the stiffness, and a reasonable choice of stiffness is important for practical applications.

To understand the performance of our method compared to other penalty-based methods, we test our contact algorithm against that of [20]. Direct and fair comparison is complicated by the fact that [20] is a fully explicit method (Runge Kutta and a contact/friction postprocess). We have implemented the contact and friction forces of [20] using our implicit integration and CCD-based collision detection (which conveniently gives us robust normals automatically). We implemented two variants of their algorithm: (1) explicit contact and friction ("prmd_dd_lag") and (2) implicit normal contact followed by explicit friction ("prmd_dd_lagfr"). The discontinuous nature of the dynamic/static friction transition prevents us from implementing a fully-implicit version of their friction force. We compare with our own with implicit contact and friction ("prmd_dd"). We use a pyramid stack of four deformable objects (like Fig. 6 but with all four objects deformable). From this test, we observed that neither (1) nor (2) leads to stable stacks; both eventually collapse (see the supplementary video). This is not particularly surprising; indeed, our method will also not get stable stacks if friction is not treated implicitly. In particular, it is not sufficient for a penalty method to use

TABLE 1
Simulation Parameters and Timings for All of Our Simulations.

Name	dt	$\frac{\text{Time}}{\text{frame}}(s)$	CCD?	μ	k	grid	#p†	Tol
stack	0.01	0.12	Y	0.3	10^6		432	1
prmd_rd	0.01	0.10	Y	0.3	10^3		507	0.1
prmd_rm	0.01	4.28	Y	0.3	10^2	32^3	11k	1
prmd_rm slip	0.01	1.02	Y	0.05	10^2	32^3	11k	1
prmd_dd	0.002	0.53	Y	0.2	10^3		676	0.1
prmd_dd_lag	0.002	0.56	Y	0.2	10^3		676	0.1
prmd_dd_lagfr	0.002	0.58	Y	0.2	10^3		676	0.1
plane_r slide	0.001	0.20	N	0.1	10^3			1
plane_r stick	0.001	0.07	N	0.125	10^3			1
plane_d slide	0.001	0.05	N	0.1	10^2		8	1
plane_d stick	0.001	0.01	N	0.125	10^2		8	0.1
iplane_m slide	0.005	6.30	N	0.1	1	32^3	1.3k	1
iplane_m stick	0.005	1.34	N	0.125	1	32^3	1.3k	1
plane_m slide	0.005	1.44	N	0.1	1	32^3	1.3k	1
plane_m stick	0.005	2.16	N	0.125	1	32^3	1.3k	1
wdg_rm	0.01	0.30	N	0.25	10^2	32^3	1.5k	1
wdg_rm fall	0.01	0.18	N	0.1	10^2	32^3	1.5k	1
wdg_dr	0.002	0.62	N	0.05	10^3		169	0.1
wdg_dr fall	0.002	8.41	N	0.002	10^3		169	0.1
wdg_rr	0.01	0.05	N	0.08	10			1
wdg_rr fall	0.01	0.03	N	0.03	10			1
wdg_dr_r	0.01	1.26	N	0.03	10^3		169	0.1
plane_sph roll	0.005	0.05	Y	0.3	10			1
plane_sph slip	0.005	0.04	Y	0.01	10			1
torus_dr	0.002	2.17	Y	0.6	10^2		2.6k	0.01
cloth	0.005	6.73	Y	0.3	500		3.8k	1
sandbox_cube	0.001	242^+	Y	0.3	10^5	64^3	0.4m	1
sandbox_sph	0.001	7095^+	N	0.9	25000	128^3	0.9m	0.1
sand_on_sph	0.001	597	Y	0.9	10^4	96^3	64k	1
goo_on_sph	0.001	147^+	Y	0.3	10^4	96^3	120k	1
bowl	0.001	632	Y	0.3	10^5	96^3	31k	0.01

† Number of particles used in a test.
+ The timing information is measured using 8 threads.

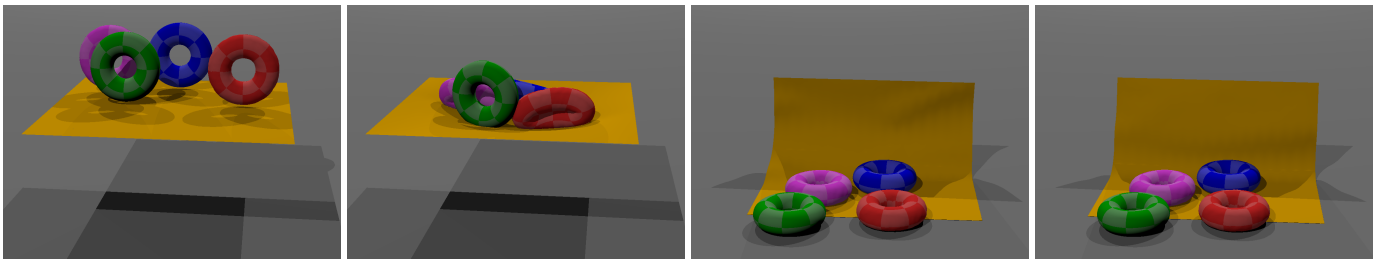


Fig. 15. Two deformable and two rigid bodies are dropped onto a piece of cloth.

attachments to achieve stable stacking.

We also examined the runtime of the two approaches, noting that we are running [20] under conditions for which it was never intended (implicit integration) and with a collision detection scheme that is likely less efficient than the non-CCD library that the original method used. Neither version was optimized. We observe that version (2) is slightly slower than our version. Since (2) uses implicit normal contact, CCD is performed in each Newton iteration as it is with our method. Since the actual force computations are negligible for both methods, we would expect the two to run at about the same pace, and indeed the performance is very close (see Table 1). There is a slight extra cost to (2), however, and it is caused by degradation in the performance of the Newton solver. In our simulation, our stack settles down, which allows the Newton solver to converge somewhat quicker. Lagging contact and/or friction means contact and elasticity are always fighting, which slows down the Newton solver. The comparison of (1) with our method is somewhat more surprising. In this case, contact and friction are both explicit, and CCD is performed only once per time step. This reduces the cost of CCD (from 32% for our method to 5% for (1)). This savings, however, comes at a cost. The fighting between elastic and contact forces is now much worse (before elastic forces only fought with friction). The resulting slower convergence in Newton's method is greater than the savings from CCD, and (1) ends up also running slower than our scheme. We note, however, that the performance differences are very slight, and which version is faster will depend strongly on the example setup, collision detection scheme chosen, level of code optimization, and a large number of other factors. Our contact algorithm is not intended as a way to make contact faster; rather, this comparison merely shows that it is competitive with existing methods and not too slow to be practical.

5 Conclusions and Future Work

We have demonstrated that we are able to simultaneously apply general elastic forces and frictional contact between different types of objects through the use of a novel frictional contact penalty force. We have demonstrated that this force allows us to achieve MPM-rigid and deformable-rigid coupling. Our force enables us to enforce frictional boundary conditions against the particles of an MPM simulation rather than being limited to processing these contacts on grid nodes, which prevents particle drift or bunching at collision objects.

Although the method is quite versatile, it has several important limitations. Because contact is enforced with a penalty force, it adds extra stiffness to the system and does not completely prevent penetration between objects. As is normally the case for penalty forces, our penalty force does not model restitution. One promising avenue for extending our method to include a coefficient of restitution is through the use of a nonlinear damped spring [19]. Our implementation assumes isotropic friction; we see no reason that anisotropic friction could not be implemented by looking up local friction parameters (at the attachment location) and then replacing the circular cone with an elliptical cone in the yield criterion.

When colliding triangle meshes, we only process point-triangle pairs. In particular, we do not consider edge-edge pairs. We made this compromise for practical reasons. Using an edge in place of a particle introduces many complications. For example, would the edge be connected to an attachment point or an attachment edge? Is the attachment at the original barycentric collision location, or can the spring force slide along the edge? If it slides to the edge, does it become a point collision again?

The linear systems that result are asymmetrical, and we must solve them with GMRES. Although the memory requirements and computational load of GMRES scale quadratically with the number of iterations, our stabilization of the Newton solve safely allows us to limit the number of iterations (we use 20). If our Newton step was not computed accurately, we can rely on the line search.

Another limitation is that we can only hope to converge to a local minimum of our objective E. Unlike more standard optimization formulations, a local minimum for our objective does not lead to a solution to Newton's second law, though we have never observed it to converge to a non-global minimum. Our line searches are also more sensitive to the force derivatives than normal, since the derivatives used by the Wolfe line search involve the force derivatives; only forces are involved in the standard formulation. The level set formulation is limited to smooth objects due to the presence of sonic points; the CCD formulation is smoother and does not have sonic points. The CCD formulation presented is not free from kinks and discontinuities, however, and we have observed Newton's method to fail to converge for the CCD formulation. This normally occurs when the system is nearly converged (so that the objective slope is low) and when the penalty stiffness is high (so that any kinks or discontinuities that may be present are amplified). Since the system is usually near convergence anyway, we simply continue the simulation with the incompletely converged result. We leave the problem of developing a more kink-resistant solver formulation for future work.

Our method is not as efficient as less strongly coupled formulations (such as ones that lag friction) due to the extra stiffness and asymmetric problem. Timing results and

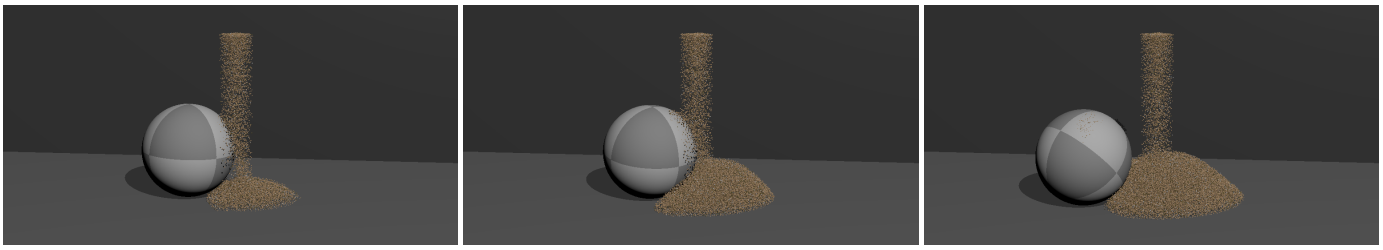


Fig. 16. We pour sand over a sphere; friction between the sand and the sphere causes the sphere to be pulled into the sand column

parameters for all of our simulations are given in Table 1.

ACKNOWLEDGMENTS

We would like to thank Tamar Shinar for providing helpful suggestions. We would also like to thank Jonathan Kaneshiro for helping with rendering.

REFERENCES

[1] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '98. New York, NY, USA: ACM, 1998, pp. 43–54.

[2] T. F. Gast, C. Schroeder, A. Stomakhin, C. Jiang, and J. M. Teran, "Optimization integrator for large time steps," *IEEE transactions on visualization and computer graphics*, vol. 21, no. 10, pp. 1103–1115, 2015.

[3] E. Coevoet, A. Escande, and C. Duriez, "Optimization-based inverse model of soft robots with contact handling," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1413–1419, 2017.

[4] D. Baraff, "Coping with friction for non-penetrating rigid body simulation," *ACM SIGGRAPH computer graphics*, vol. 25, no. 4, pp. 31–41, 1991.

[5] R. W. Cottle, J.-S. Pang, and R. E. Stone, *The Linear Complementarity Problem*. Academic Press, 1992.

[6] D. E. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction," *International Journal for Numerical Methods in Engineering*, vol. 39, no. 15, pp. 2673–2691, 1996.

[7] M. Anitescu and F. A. Potra, "Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems," *Nonlinear Dynamics*, vol. 14, no. 3, pp. 231–247, 1997.

[8] M. A. Otaduy, R. Tamstorf, D. Steinemann, and M. Gross, "Implicit contact handling for deformable objects," in *Computer Graphics Forum*, vol. 28, no. 2. Wiley Online Library, 2009, pp. 559–568.

[9] E. Drumwright, "A fast and stable penalty method for rigid body simulation," *IEEE transactions on visualization and computer graphics*, vol. 14, no. 1, pp. 231–240, 2008.

[10] H. Courtecuisse, J. Allard, P. Kerfriden, S. P. Bordas, S. Cotin, and C. Duriez, "Real-time simulation of contact and cutting of heterogeneous soft-tissues," *Medical image analysis*, vol. 18, no. 2, pp. 394–410, 2014.

[11] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically deformable models," *ACM Siggraph Computer Graphics*, vol. 21, no. 4, pp. 205–214, 1987.

[12] M. McKenna and D. Zeltzer, "Dynamic simulation of autonomous legged locomotion," in *ACM SIGGRAPH Computer Graphics*, vol. 24, no. 4. ACM, 1990, pp. 29–38.

[13] M. Moore and J. Wilhelms, "Collision detection and response for computer animation," in *ACM Siggraph Computer Graphics*, vol. 22, no. 4. ACM, 1988, pp. 289–298.

[14] S. Hasegawa, N. Fujii, K. Akahane, Y. Koike, and M. Sato, "Real-time rigid body simulation for haptic interactions based on contact volume of polygonal objects," *Transactions of the Society of Instrument and Control Engineers*, vol. 40, no. 2, pp. 122–131, 2004.

[15] X. Provot, "Collision and self-collision handling in cloth model dedicated to design garments," in *Computer Animation and Simulation '97*. Springer, 1997, pp. 177–189.

[16] R. Bridson, R. Fedkiw, and J. Anderson, "Robust treatment of collisions, contact and friction for cloth animation," *ACM Transactions on Graphics (ToG)*, vol. 21, no. 3, pp. 594–603, 2002.

[17] E. Guendelman, R. Bridson, and R. Fedkiw, "Nonconvex rigid bodies with stacking," in *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3. ACM, 2003, pp. 871–878.

[18] Y. Hwang, E. Inohira, A. Konno, and M. Uchiyama, "An order n dynamic simulator for a humanoid robot with a virtual spring-damper contact model," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 1. IEEE, 2003, pp. 31–36.

[19] D. W. Marhefka and D. E. Orin, "Simulation of contact using a nonlinear damping model," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 2. IEEE, 1996, pp. 1662–1668.

[20] K. Yamane and Y. Nakamura, "Stable penalty-based model of frictional contacts," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 1904–1909.

[21] H. Xu, Y. Zhao, and J. Barbič, "Implicit multibody penalty-based distributed contact," *IEEE transactions on visualization and computer graphics*, vol. 20, no. 9, pp. 1266–1279, 2014.

[22] S. Fisher and M. C. Lin, "Fast penetration depth estimation for elastic bodies using deformed distance fields," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 1. IEEE, 2001, pp. 330–336.

[23] M. Tang, D. Manocha, M. A. Otaduy, and R. Tong, "Continuous penalty forces." *ACM Trans. Graph.*, vol. 31, no. 4, pp. 107–1, 2012.

[24] D. Baraff and A. Witkin, "Partitioned dynamics," CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, Tech. Rep., 1997.

[25] J. F. O'Brien, V. B. Zordan, and J. K. Hodgins, "Combining active and passive simulations for secondary motion," *IEEE Computer Graphics and Applications*, vol. 20, no. 4, pp. 86–96, 2000.

[26] J. Jansson and J. S. Vergeest, "Combining deformable-and rigid-body mechanics simulation," *The Visual Computer*, vol. 19, no. 5, pp. 280–290, 2003.

[27] J. Lenoir and S. Fonteneau, "Mixing deformable and rigid-body mechanics simulation," in *Computer Graphics International, 2004. Proceedings*. IEEE, 2004, pp. 327–334.

[28] E. Sifakis, T. Shinar, G. Irving, and R. Fedkiw, "Hybrid simulation of deformable solids," in *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, 2007, pp. 81–90.

[29] J. Kim and N. S. Pollard, "Fast simulation of skeleton-driven deformable body characters," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 5, p. 121, 2011.

[30] T. Shinar, C. Schroeder, and R. Fedkiw, "Two-way coupling of rigid and deformable bodies," in *SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2008, pp. 95–103.

[31] Y. Yang, G. Rong, L. Torres, and X. Guo, "Real-time hybrid solid simulation: spectral unification of deformable and rigid materials," *Computer Animation and Virtual Worlds*, vol. 21, no. 3-4, pp. 151–159, 2010.

[32] D. Sulsky, Z. Chen, and H. Schreyer, "A particle method for history-dependent materials," *Comp Meth in App Mech Eng*, vol. 118, no. 1, pp. 179–196, 1994.

[33] D. Sulsky, S. Zhou, and H. Schreyer, "Application of a particle-in-cell method to solid mechanics," *Comp Phys Comm*, vol. 87, no. 1, pp. 236–252, 1995.

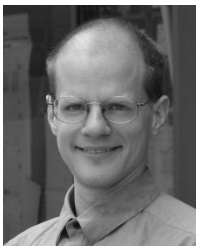
[34] A. Stomakhin, C. Schroeder, L. Chai, J. Teran, and A. Selle, "A material point method for snow simulation," in *ACM Transactions on Graphics (SIGGRAPH 2013)*, 2013, pp. 102:1–10.

[35] G. Klár, T. Gast, A. Pradhana, C. Fu, C. Schroeder, C. Jiang, and T. Teran, "Drucker-prager elastoplasticity for sand animation," *ACM Transactions on Graphics (SIGGRAPH 2016).*, 2016.

[36] G. Daviet and F. Bertails-Descoubes, "A semi-implicit material

point method for the continuum simulation of granular materials," *ACM Trans Graph*, vol. 35, no. 4, jul 2016.
[37] R. Narain, A. Golas, and M. Lin, "Free-flowing granular materials with two-way solid coupling," *ACM Trans Graph*, vol. 29, no. 6, pp. 173:1–173:10, 2010.
[38] D. M. Kaufman, S. Sueda, D. L. James, and D. K. Pai, "Staggered projections for frictional contact in multibody systems," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 5, p. 164, 2008.
[39] J. Nocedal and S. Wright, *Numerical Optimization*, ser. Springer series in operations research and financial engineering. Springer, 2006.
[40] F. Bertails-Descoubes, F. Cadoux, G. Daviet, and V. Acary, "A nonsmooth newton solver for capturing exact coulomb friction in fiber assemblies," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 1, p. 6, 2011.
[41] A. Stomakhin, R. Howes, C. Schroeder, and J. Teran, "Energetically consistent invertible elasticity," in *Proc. Symp. Comp. Anim.*, 2012, pp. 25–32.
[42] R. Bridson, S. Marino, and R. Fedkiw, "Simulation of clothing with folds and wrinkles," in *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, ser. SCA '03. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003, pp. 28–36.

Ounan Ding Ounan Ding received his B.S. in computer science in computer science at Nankai University in 2012. He is currently pursuing his Ph.D. in computer science at University of California, Riverside.



Craig Schroeder Craig Schroeder received his Ph.D. in computer science from Stanford University in 2011. After graduation, he did a postdoc in applied mathematics at UCLA, where he was awarded the Chancellor's Award for Postdoctoral Research in 2013. He is currently an assistant professor in computer science and engineering at the University of California Riverside. He collaborated with Pixar Animation Studios during his Ph.D. and with Walt Disney Animation Studios during his postdoctoral studies. For his research contributions he received screen credits in Pixar's "Up" and Disney's "Frozen." He actively publishes in both computer graphics and computational physics, and his primary areas of interest are the material point method and robust numerical methods.