To appear in the ACM SIGGRAPH 2010 conference proceedings

Volume Contact Constraints at Arbitrary Resolution

Jérémie Allard[1,2] François Faure[3,1,4] Hadrien Courtecuisse[1,2] Florent Falipou[1] Christian Duriez[1,2] Paul G. Kry[5]

[1]INRIA [2]University of Lille [3]University of Grenoble [4]LJK – CNRS [5]McGill University

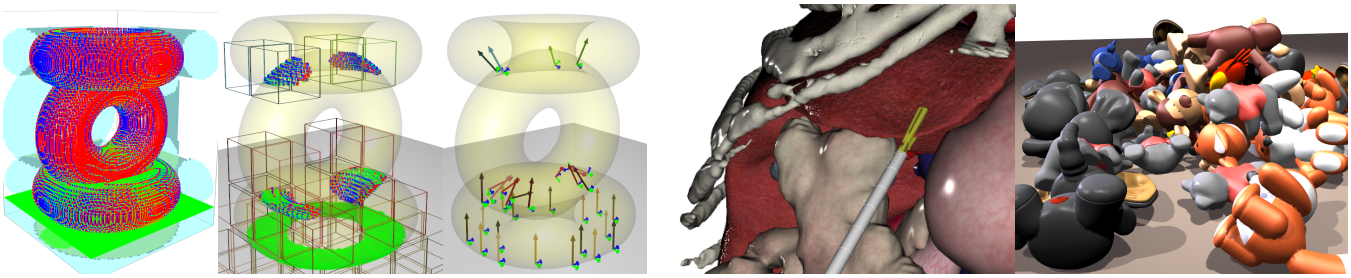


Figure 1: *Left: A stack of tori, with the intersection volumes and normals on the GPU. Right: intersection volume constraints allow us to resolve frictional contact in challenging scenarios.*

Abstract

We introduce a new method for simulating frictional contact between volumetric objects using interpenetration volume constraints. When applied to complex geometries, our formulation results in dramatically simpler systems of equations than those of traditional mesh contact models. Contact between highly detailed meshes can be simplified to a single unilateral constraint equation, or accurately processed at arbitrary geometry-independent resolution with simultaneous sticking and sliding across contact patches. We exploit fast GPU methods for computing layered depth images, which provides us with the intersection volumes and gradients necessary to formulate the contact equations as linear complementarity problems. Straightforward and popular numerical methods, such as projected Gauss-Seidel, can be used to solve the system. We demonstrate our method in a number of scenarios and present results involving both rigid and deformable objects at interactive rates.

CR Categories: I.3.5 [Computer Graphics]: Physically based modeling— [I.3.7]: Computer Graphics—Animation

Keywords: physically based animation, contact forces, Coulomb friction, constraints

1 Introduction

Dealing with contact is a fundamental problem in physically based computer animation, and possibly one of the most challenging. Accurate simulation of frictional contact is important for a wide variety of natural phenomena and has numerous applications, from video games, to robotics.

The computation of contact forces involves several important difficulties. One such problem is collision detection and modeling. This requires not only detection of the intersection of two objects, which is a complex problem on its own, but also computation of the depth and direction. While the problem can be straightforward when geometric meshes are in close proximity, or in point contact, the solution is less clear in cases that involve deep interpenetration. This imposes restrictions on the time step size, or obliges the use of continuous collision detection methods that permit the processing of collision at the exact time of contact. For interactive simulation, however, it can be preferable to use methods that can handle interpenetrations gracefully, without additional computational cost.

Another difficulty in contact force computation is the solution of the contact equations themselves. Solving contact with Coulomb friction involves Lagrange multipliers and coupled inequalities. With the inclusion of dry friction, the problem is larger and particularly complex because we must determine which constraints are active and this can be a hard combinatorial problem. Furthermore, numerous simultaneous contacts can lead to constraints that may be redundant, or even inconsistent, which results in numerical problems. Processing contacts sequentially alleviates some difficulties but may create artifacts and raises convergence issues. Despite decades of research and significant progress, the computation of contact forces between complex geometries remains a hard problem. Interactive applications use simplified collision geometries such as spheres, cylinders, and cubes, which restricts the range of simulated objects or leads to visible artifacts. The simulation of contact between fine meshes is generally performed offline.

In this paper, we propose a completely new approach for contact force computation, suitable for all objects bounded by triangular meshes. The collision detection and modeling phase returns the size of the intersection volume and its derivative with respect to the vertices of the meshes. In contrast to traditional methods, this allows the non-interpenetration constraint to be modeled using a single scalar equation, *i.e.*, the size of the intersection volume, rather than a (potentially large) number of distance constraints between geometric primitives. The result is a dramatically simpler system of equations. Figure 1 shows a visualization of this and a preview of our results. Effectively, this approach can be seen as using *contact volumes* as opposed to *contact points* to formulate the problem. Additionally, we show how to constrain the relative tangential motion, allowing frictional contact between arbitrarily complex objects. To allow simultaneous sticking and sliding behavior across the con-

tacts between objects, we extend our geometric model by splitting the single contact volume into several contacts. We use a regular grid where the contact equations are generated from the cells that contain parts of the intersection volume. Tuning the resolution of the grid allows a trade off between accuracy and speed. To the best of our knowledge, this is the first method where the complexity of the contact model is completely independent from the complexity of the surface model without pre-processing.

Our contact equations are purely geometric and can be applied to arbitrary Lagrangian models including rigid and deformable solids. They can be solved using the traditional numerical methods. Moreover, no precomputation is needed, which makes the method suitable for animations with topology changes such as cutting, fracturing, or adaptive levels of detail. See Figure 2 for an example of this and other difficult scenarios that our method handles efficiently. Finally, a major advantage of our method is that it is no longer necessary to worry about collision proxies, signed distance fields, or other adapted collision models; collision processing and contact force computation can easily be done with the same highly detailed geometry used for rendering.

The remainder of this paper is organized as follows. We review related work in Section 2 and provide the necessary background in Section 3. Our new constrained volume method is presented in Section 4. This is extended to multiple volumes in Section 5. Results are presented and discussed in Section 6.

2 Related Work

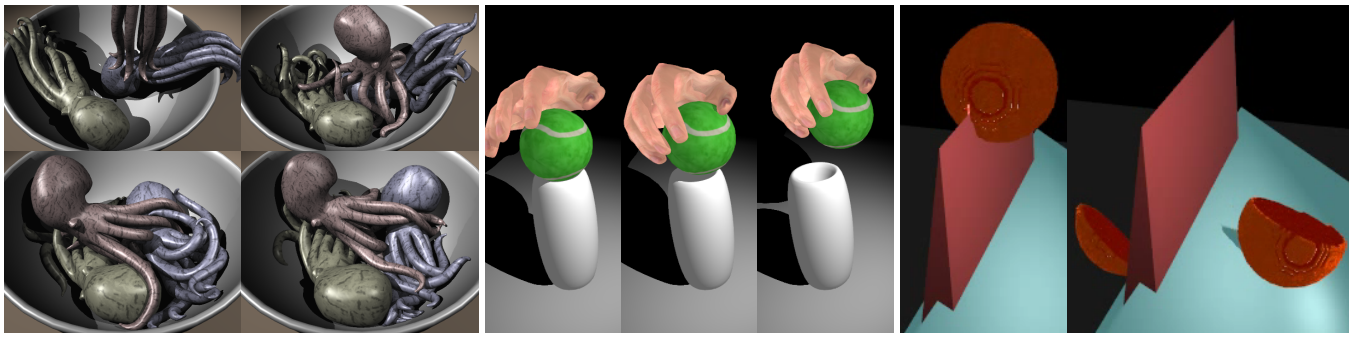
In most previous work, the problem of collision detection and response are addressed separately. The amount of work done on collision detection alone is quite vast. Collision detection typically involves hierarchical bounding volume data structures, such as bounding boxes [Gottschalk et al. 1996] or spheres [Hubbard 1995]. In addition to rigid object collision, significant work has been done to address deformable models [van den Bergen 1997; James and Pai 2004], and self-contact [Volino and Magnenat-Thalmann 1995; Provot 1997]. See the survey by Teschner et al. [2004] for an excellent overview of different methods. More recently, techniques which exploit the GPU have been proposed as this can result in important speed improvements through parallelization [Vassilev et al. 2001; Heidelberger et al. 2003; Heidelberger et al. 2004; Baciu and Wong 2004; Wong and Baciu 2005; Sud et al. 2006]. Most of the techniques return pairs of primitives, between which distance constraints can be formulated to resolve the contacts. Otaduy et al. [2004] use the GPU to compute penetration depth in a given direction. Our work builds on that of Heidelberger et al. extended by Faure et al. [2008], where the GPU is used to return the volume of intersection and self-intersection between the objects, along with the derivatives of these volumes. This new type of collision data allows a new approach of collision response. Our work significantly improves upon that of Faure et al. in that we show how to formulate unilateral contact constraints instead of penalty forces, Coulomb friction instead of viscous friction, and we introduce an arbitrary-resolution formulation for resolving varying interaction forces across contact patches.

While it is possible to solve for contact response using penalty methods, it has long been recognized that there are advantages to formulating contact with constraints. For contact, the constraints are unilateral, which results in a linear complementarity problem (LCP). The addition of Coulomb friction involves a non-linear constraint relating the tangential force to the normal force. Most work in graphics uses a pyramid discretization of the friction cone to formulate the problem as an LCP [Baraff 1991; Milenkovic and Schmidl 2001], but it is also possible to use the exact cone (a non-linear complementarity problem) and to compute the solution using iterative methods [Duriez et al. 2006]. While natural to formulate the frictional contact equations at the acceleration level, a velocity level formulation is preferred since it avoids the Painlevé paradox [Stewart 2000; Stewart and Trinkle 1996; Anitescu and Potra 1997]. However, the problem can still result in an NP-hard combinatorial problem to identify the active constraints [Baraff 1991]. While some recent work strives to find solutions which are as correct as possible [Harmon et al. 2009; Kaufman et al. 2008], we instead focus on an approximate solution that is fast and interactive. We exploit small interpenetrations to formulate constraints, which also makes our approach naturally robust to larger interpenetrations. Velocity level implicit integration is typical in many works in graphics because it can also lead to stable simulations with large time steps [Baraff and Witkin 1998]. Otaduy et al. [2009] use velocity level implicit contact constraints in combination with iterative methods to simulate combined elastic, rigid, and thin shells. Velocity level LCPs have also been used for fluid-solid coupling [Batty et al. 2007]. Other relevant work on friction includes fast approximate models in video game systems [Parker and O'Brien 2009], and simulation of anisotoripic friction [Pabst et al. 2009].

In graphics, the methods used to solve contact equations can be classified into two categories: iterative methods that repeatedly process the contacts sequentially [Erleben 2007; Duriez et al. 2006; Otaduy et al. 2009], and direct methods that build and solve a single system of equations [Baraff 1991; Baraff 1994; Pauly et al. 2004]. Kaufmann et al. [2008] use a direct solver, but alternatively solve nonpenetration and friction constraints to avoid scaling and non-convexity problems associated with traditional direct methods, while improving on the slow convergence that can be typical with iterative methods. Alternatively, for frictional dynamics, Lemke's direct method can be modified to run faster by introducing frictional complementary constraints only when necessary [Lloyd 2005]. But much of the recent work in graphics has shifted to iterative methods because the implementation of such methods is less complex and the computation time is easier to control. Our method can be applied to both approaches. In this work we use iterative methods, and our arbitrary-resolution frictional contact formulation helps address the problem of slow convergence by decoupling the resolution of the contact from that of the geometry.

Models at multiple resolutions have been exploited in graphics for simulation of elastic materials and deformation of models [Debunne et al. 2001; Otaduy et al. 2007; Barbič and James 2007; Shi et al. 2006; Müller 2008]. The main contributions in previous work have been irregular nesting, haptic rates for frictionless contact, and speed improvements through the use of multigrid. Our work takes the unique approach of treating the contact constraints at an adjustable resolution, independent of the degrees of freedom. While there has been vast amounts of work on physically based deformation (see Nealen et al. [Nealen et al. 2005] for a good survey), the focus is typically deformation as opposed to contact.

Numerical integration of differential equations with position level constraints requires stabilization to avoid numerical drift [Ascher and Petzold 1998]. If small steps are taken then drift is not always a significant problem and can be left as a separate problem [Kaufman et al. 2005; Kaufman et al. 2008]. Gundleman et al. [] provide a bottom up approach to deal with stacking. A related problem is the robust treatment of contact resolution [Bridson et al. 2002; Harmon et al. 2008]. Our work uses larger step sizes, and both allows and exploits interpenetration, thus, we also include stabilization. Baumgarte [1972] introduced a popular stabilization method, which is widely used, but also widely criticized for the difficulty of tuning parameters. Post stabilization is a preferable alternative [Ascher and Petzold 1998] and is straightforward to implement as the constraint gradient is available from the formulation of the constrained



(a) *Complex collisions and self-collisions* **(b)** *Grasping with stable static friction* **(c)** *Cutting an object with dynamic collision surfaces*

Figure 2: *Demonstration of several difficult scenarios for contact modeling that are efficiently handled by the proposed method.*

equations of motion. However, unilateral constraints should not be stabilized in the same manner as bilateral constraints. A separate LCP can be formed to solve the post step [Cline and Pai 2003], or in some cases the time stepping and unilateral post step can be combined into a single LCP [Anitescu and Hart 2004]. While our formulation can accommodate any of these approaches to stabilization, our solution most closely resembles that of Cline and Pai [2003] as we solve an iterative LCP to perform the position correction step.

3 Background

In this section we introduce the necessary background material on which we build our method. This consists of volume and volume-gradient computation from layered depth images (LDIs), and a review of constrained dynamics. We consider a physical system governed by the ordinary differential equation

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{f}_{\text{ex}} \tag{1}$$

where vector $\mathbf{q}$ is the independent degrees of freedom (DOFs), $\mathbf{M}$ is the mass Matrix, $\mathbf{f}$ provides the internal forces, and $\mathbf{f}_{\text{ex}}$ are the external forces (such as those due to contact and friction).

3.1 Numerical Integration and Contact Constraints

Overall, the methods that we describe here have also been used or described in depth with small variations by many others [Baraff and Witkin 1998; Erleben 2007; Duriez et al. 2006; Otaduy et al. 2009; Kaufman et al. 2008; Pabst et al. 2009; Parker and O'Brien 2009].

Integration We use an implicit integration method to solve the ODE in Equation 1 because our system may involve stiff elastic deformation (the velocity level formulation will also be of benefit for friction). To reduce computation time, we use a backward Euler scheme with one Newton iteration to solve for the state update [Baraff and Witkin 1998]. This works well as the internal forces are often smooth, but we note that other numerical integration methods with multiple Newton iterations could also be used for better accuracy at additional cost.

Given a time step size h and the current state $(\mathbf{q}_0, \dot{\mathbf{q}}_0)$, we compute an implicit velocity update $\Delta\dot{\mathbf{q}}$ by solving the linearized system

$$\mathbf{A}\Delta\dot{\mathbf{q}} = \mathbf{b}, \tag{2}$$

where $\mathbf{A} = \mathbf{M} - h\mathbf{B} - h^2\mathbf{K}$, and $\mathbf{b} = h\mathbf{f}(\mathbf{q}_0, \dot{\mathbf{q}}_0) + h^2\mathbf{K}\dot{\mathbf{q}}_0$. Recall, $\mathbf{B} = \frac{\partial \mathbf{f}}{\partial \dot{\mathbf{q}}}$ and $\mathbf{K} = \frac{\partial \mathbf{f}}{\partial \mathbf{q}}$ are the damping and stiffness matrices. From the solution, the velocities at the next time step are computed as

$$\dot{\mathbf{q}}_{0+h} = \dot{\mathbf{q}}_0 + \Delta\dot{\mathbf{q}}. \tag{3}$$

These new velocities are subsequently used to perform an implicit update on the position, *i.e.*, $\mathbf{q}_{0+h} = \mathbf{q}_0 + h\dot{\mathbf{q}}_{0+h}$, or using the exponential map for rigid motion.

Contact When point contacts exist in the system, the contacts form separation distance constraints, $\mathbf{g}(\mathbf{q}) \geq 0$. The transpose of the constraint gradient $\mathbf{J} = \partial\mathbf{g}/\partial\mathbf{q}$ provides the contact force directions (*i.e.*, contact normals), which we add to the system using Lagrange multipliers $\boldsymbol{\lambda}$. Thus, Equation 2 becomes

$$\mathbf{A}\Delta\dot{\mathbf{q}} = \mathbf{b} + \mathbf{J}^T\boldsymbol{\lambda}. \tag{4}$$

The solution of this system must respect the Signorini condition. That is, contacts do not produce attraction forces, $\boldsymbol{\lambda} \geq 0$, separation distance at contacts must remain non-negative, $\mathbf{g}(\mathbf{q}) \geq 0$, and the constraint is active ($\boldsymbol{\lambda}$ nonzero) if and only if the separation distance is zero. We write this as $0 \leq \mathbf{g}(\mathbf{q}_{0+h}) \perp \boldsymbol{\lambda} \geq 0$, and note that we must differentiate the constraint to produce complementarity conditions involving only the unknown velocity update and Lagrange multipliers,

$$0 \leq \mathbf{J}(\dot{\mathbf{q}}_0 + \Delta\dot{\mathbf{q}}) \perp \boldsymbol{\lambda} \geq 0. \tag{5}$$

The combination of Equations 4 and 5 produces a mixed LCP (MLCP), which can be solved with either direct or iterative methods (see discussion in Section 2).

Stabilization Discrete collision detection can not anticipate the apparition of new contacts, and numerical integration errors can produce drift. Thus, a post stabilization position update is computed at the end of each numerical integration step. However, the position update $\Delta\mathbf{q}$ must also respect the complementarity conditions. Either there is a separation at a contact, $\mathbf{g}(\mathbf{q}_{0+h}) > 0$, or the constraint provides a repulsive force, $\boldsymbol{\zeta} > 0$, but not both. This produces the MLCP

$$\mathbf{A}\Delta\mathbf{q} = \mathbf{J}^T\boldsymbol{\zeta}, \tag{6}$$

$$0 \leq \mathbf{g}(\mathbf{q}_0) + \mathbf{J}\Delta\mathbf{q} \perp \boldsymbol{\zeta} \geq 0, \tag{7}$$

which is solved for $\boldsymbol{\zeta}$ and $\Delta\mathbf{q}$. When $\mathbf{g}$ is smooth, one iteration of this linearized post step is sufficient to produce a good correction.

Friction Coulomb friction involves additional tangential forces at the contact points. These are typically written in a symmetric basis of 4 vectors at each contact point (negative and positive $\mathbf{t_1}$ and $\mathbf{t_2}$ directions for each contact coordinate frame $\{\mathbf{n}, \mathbf{t_1}, \mathbf{t_2}\}$), which lets the friction force be written as a combination involving only positive weights. For contact i, let $0 \leq \boldsymbol{\beta}_i \in \mathbb{R}^4$ be the coordinates

of the friction force and $\mathbf{T}_i^T$ be the basis that applies the force on the DOFs in equal and opposite directions. Equation 4 becomes

$$\mathbf{A}\Delta\dot{\mathbf{q}} = \mathbf{b} + \mathbf{J}^T\boldsymbol{\lambda} + \sum \mathbf{T}_i^T\boldsymbol{\beta}_i, \tag{8}$$

but it is now subject to additional complementarity conditions. Specifically, there is either tangential sliding at the contact, or the contact force is not on the boundary of the friction cone, but not both. For contact i, the pyramidal friction cone conditions can be written

$$0 \leq \sigma_i \perp \mu\lambda_i - e\boldsymbol{\beta}_i \geq 0, \tag{9}$$

$$0 \leq \boldsymbol{\beta}_i \perp \mathbf{T}_i^T(\dot{\mathbf{q}}_0 + \Delta\dot{\mathbf{q}}) - e^T\sigma_i \geq 0, \tag{10}$$

where $e = (1, 1, 1, 1)$, and the slack variable σ_i is nonzero when there is sliding at the contact. One only needs the contact normals to build a tangent space basis to set up these frictional contact equations. Note also that the friction constraint is not considered in the post stabilization step.

3.2 Volume-Based Penalty Force

Let us briefly review the image-based collision response method we extend in this work. The basic idea is to minimize the intersection volume between two polyhedra. The intersection volume is computed on the GPU based on bounding pixels in the rasterization of the object surfaces into Layered Depth Images (LDIs) [Heidelberger et al. 2003; Heidelberger et al. 2004]. This data structure uses a stack of images to represent the object sufaces as discrete height fields. The necessary number of images depends on the number of surface layers in the chosen viewing direction. Each pixel stores the surface depth in the viewing direction, as well as additional data such as normal orientation or object index. By sorting the depths stored in the different images at each pixel (i, j), we obtain the ordered list of surface intersections with a ray parallel to the viewing direction. Along the ray, each object volume is represented by one or several depth intervals between entry points and exit points. Collision detection is straightforward based on interval intersections. One LDI in an arbitrary viewing direction is sufficient to detect collisions between volumetric objects. Computing the gradient of the intersection volume requires three LDIs in mutually orthogonal directions [Faure et al. 2008]. At left in Figure 1 is a depiction of LDI volume models using red, green, and blue pixels to denote x, y, and z viewing directions, respectively. The corresponding intersection volumes are shown in the second image of the same figure. Figure 3 represents a slice of an intersection volume. The rasterization can be done in any direction, but for simplicity we assume an orthogonal projection along one of the primary axes. Thus, the volume computed using a z projection is

$$\mathcal{V} = a \sum_{(i,j)\in\mathcal{S}_z^+} \mathrm{z}_{ij}^+ \; - \; a \sum_{(i,j)\in\mathcal{S}_z^-} \mathrm{z}_{ij}^-, \tag{11}$$

where a is the area of a pixel, z_{ij}^+ and z_{ij}^- are the upper and lower pixel depths, and the sets $\mathcal{S}_z^+$ and $\mathcal{S}_z^-$ respectively contain the pixel locations (i, j) of the upper and lower contact surfaces.

The derivative of the volume with respect to the depth of a given triangle vertex k represents the variation of the volume size corresponding to a unit displacement of the vertex in the viewing direction,

$$\frac{\partial\mathcal{V}}{\partial\mathbf{p}_k^z} = a \sum_{(i,j)\in\mathcal{S}_z^+} \frac{\partial\mathrm{z}_{ij}^+}{\partial\mathbf{p}_k^z} \; - \; a \sum_{(i,j)\in\mathcal{S}_z^-} \frac{\partial\mathrm{z}_{ij}^-}{\partial\mathbf{p}_k^z}. \tag{12}$$

where $\mathbf{p}_k^z$ is the z coordinate of vertex number k. Conveniently, the scalar $\partial\mathrm{z}_{ij}(\mathbf{p})/\partial\mathbf{p}_k^z$ at a given pixel corresponds to the barycentric coefficient used to interpolate the depth value from the depth at vertex k (that is, the value is readily available in the GPU during rasterization). If the vertex $\mathbf{p}_k$ is not part of the triangle containing the pixel then the partial derivative is zero. For most vertices, the gradient receives a contribution from at most one contact surface, either $\mathcal{S}_z^+$ or $\mathcal{S}_z^-$, but some vertices receive contributions of both, such as $\mathbf{p}_3$ in Figure 3.

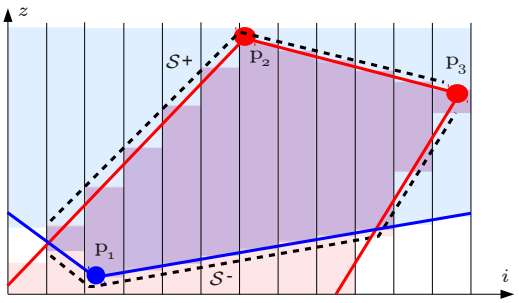


Figure 3: *A 2D slice of an LDI showing the intersecting object volumes of two objects. Here, the LDI viewing direction is the z axis. Vertices are labeled $\mathbf{p}$, while pixels are shown with horizontal lines in different columns. The intersection volume appears in purple, and is bounded by sets of surface pixels, $\mathcal{S}^+$ on top, and $\mathcal{S}^-$ on bottom (dotted lines).*

Given only a projection in z, it is not possible to use the barycentric weights to accurately compute the other partials, $\partial\mathrm{z}_{ij}(\mathbf{p})/\partial\mathbf{p}_k^x$ and $\partial\mathrm{z}_{ij}(\mathbf{p})/\partial\mathbf{p}_k^y$. Thus, we use LDIs with projections along the other two axes to compute derivatives with respect to the other vertex coordinates. For each axis, the sums over pixels for the volume and gradient computation are done simultaneously. This is done three times for the three LDIs. That is, the volume is accumulated three times (and subsequently corrected by $\frac{1}{3}$), while its derivative with respect to the coordinate of each vertex is accumulated into the gradient vector,

$$\frac{\partial\mathcal{V}}{\partial\mathbf{p}} = \left(\frac{\partial\mathcal{V}}{\partial\mathbf{p}_1} \cdots \frac{\partial\mathcal{V}}{\partial\mathbf{p}_n}\right), \tag{13}$$

$$\frac{\partial\mathcal{V}}{\partial\mathbf{p}_k} = \left(\frac{\partial\mathcal{V}}{\partial\mathbf{p}_k^x} \frac{\partial\mathcal{V}}{\partial\mathbf{p}_k^y} \frac{\partial\mathcal{V}}{\partial\mathbf{p}_k^z}\right), \tag{14}$$

where x, y, z are the three successive LDI viewing directions.

Faure et al. [2008] apply a repulsion force derived from a volume-based potential, $E = \frac{1}{2}k\mathcal{V}^2$, *i.e.*, a soft constraint. Thus, the force on vertex $\mathbf{p}_i$ is the transpose of

$$-\frac{\partial E}{\partial\mathbf{p}_i} = -k\mathcal{V}\frac{\partial\mathcal{V}}{\partial\mathbf{p}_i} \tag{15}$$

In this paper, we exploit the volume and its gradient differently, but they remain the only necessary geometric values.

When the independent DOFs $\mathbf{q}$ are not directly the vertex positions $\mathbf{p}$, as in the case of rigid bodies or detailed surfaces embedded in coarse deformable models, we use the chain rule to differentiate the intersection volume with respect to the DOFs: $\frac{\partial\mathcal{V}}{\partial\mathbf{q}} = \frac{\partial\mathcal{V}}{\partial\mathbf{p}}\frac{\partial\mathbf{p}}{\partial\mathbf{q}}$.

4 Volume-Based Contact Constraints

In this section, we present a new way of handling contact and friction by defining constraints based on the intersection volume and volume derivative. We first reformulate the Signorini condition in order to obtain an equivalent complementarity problem between contact reaction pressures and interpenetration volume. This

volume-based formulation can directly use the LDI collision output to define a very efficient constraint-based collision response. In Section 4.2 we address the subtleties of also adapting the Coulomb friction complementarity conditions to the volume-based formulation. In Section 4.3 we describe how these volume-based constraints are used in the main animation loop.

4.1 Signorini Condition for Volume Contacts

For simplicity, we consider a system consisting of two colliding objects, where the intersection volume $\mathcal{V}(\mathbf{q})$ is computed as described in Section 3.2. As stated in the background, a common way of dealing with contact between two objects is to build a complementarity problem between separation distances $\mathbf{g}$ and reaction force magnitudes $\boldsymbol{\lambda}$. If two deformable objects are in contact over an area, then this involves using multiple contact constraints to track the separation distances across the contact patch.

Equivalence Recall that each unilateral point-contact constraint has conditions $0 \leq \mathbf{g}_i(\mathbf{q}_{0+h}) \perp \boldsymbol{\lambda}_i \geq 0$. The solution of the constrained system must respect the principle of virtual work. That is, constraint satisfaction must not add or remove energy from the system. Since we are solving the system at a velocity level, this can be written $\boldsymbol{\lambda}_i \cdot \mathbf{J}_i(\dot{\mathbf{q}}_0 + \Delta\dot{\mathbf{q}}) = 0$, where $\mathbf{J}_i = \frac{\partial \mathbf{g}_i}{\partial \mathbf{q}}$ is the separation distance gradient. However, we can associate a small area $\mathcal{A}_i$ with each contact i, and observe that

$$\boldsymbol{\lambda}_i \frac{1}{\mathcal{A}_i} \cdot \mathcal{A}_i \mathbf{J}_i(\dot{\mathbf{q}}_0 + \Delta\dot{\mathbf{q}}) = 0,$$

where $\rho_i \equiv \boldsymbol{\lambda}_i \frac{1}{\mathcal{A}_i}$ is a pressure and $\frac{\partial \mathcal{V}_i}{\partial \mathbf{q}} \equiv \mathcal{A}_i \mathbf{J}_i$ is a volume gradient. Because the inequalities of the separation distance constraint still apply, we have equivalent complementarity conditions for the pressure and volume-based constraint formulation,

$$0 \leq \rho_i \perp \frac{\partial \mathcal{V}_i}{\partial \mathbf{q}}(\dot{\mathbf{q}}_0 + \Delta\dot{\mathbf{q}}) \geq 0.$$

Approximation Now, instead of using many point-contact constraints, we use the total intersection volume to constrain the system and a single Lagrange multiplier ρ to compute the pressure acting on the contact patch. The intersection volume $\mathcal{V}(\mathbf{q})$ identifies the active constraints, its gradient can be seen as the sum of $-\mathcal{A}_i \mathbf{J}_i$ of active constraints, and the forces on the DOFs due to ρ act in the direction of this volume gradient. We call this approximation the *mono-volume* contact constraint.

Note that the mono-volume approach computes a uniform pressure across the whole contact area, which is not always realistic. This limitation is addressed in Section 5, which proposes a *multi-volume* approach as the equivalence discussed above need not be reduced to only a single volume. Nevertheless, when solving our volume-based constraints, some important physical properties are guaranteed:

- the interpenetration volume cannot increase;
- the pressure can only act to push objects apart;
- the pressure acting on the contact surface is non-zero if and only if the interpenetration volume is not decreasing.

Stabilization It is also important to note that LDI is a discrete collision detection. In practice, we can only create our volume-based constraints if there is a non-zero intersection volume. If the intersection is zero, then there is simply no contact. While the intersection volume should theoretically always be zero, it is instead allowed to remain below some small amount of interpenetration volume $\epsilon_{\mathcal{V}}$ to permit continuous contact. Thus, we modify the complementarity conditions in our volume-based formulation. For the implicit velocity update, the MLCP of Equations 4 and 5 becomes

$$\mathbf{A}\Delta\dot{\mathbf{q}} = \mathbf{b} + \mathbf{J}_{\mathcal{V}}^T \rho, \tag{16}$$

$$0 \leq \mathbf{J}_{\mathcal{V}}(\dot{\mathbf{q}}_0 + \Delta\dot{\mathbf{q}}) \perp \rho \geq 0, \tag{17}$$

where $J_{\mathcal{V}} = \frac{\partial \mathcal{V}}{\partial \mathbf{q}}$. For the post stabilization correction, the MLCP of Equations 6 and 7 becomes

$$\mathbf{A}\Delta\mathbf{q} = \mathbf{J}_{\mathcal{V}}^T \xi, \tag{18}$$

$$-\epsilon_{\mathcal{V}} \leq -\mathcal{V}(\mathbf{q}_0) + \mathbf{J}_{\mathcal{V}}\Delta\mathbf{q} \perp \xi \geq 0. \tag{19}$$

The value of $\epsilon_{\mathcal{V}}$ is discussed in section 4.3. Notice that we are effectively using the equations given in Section 3 with $g = -\mathcal{V}$, but with modifications to the position correction (as given in Equation 19) and the friction conditions (as described below in Section 4.2).

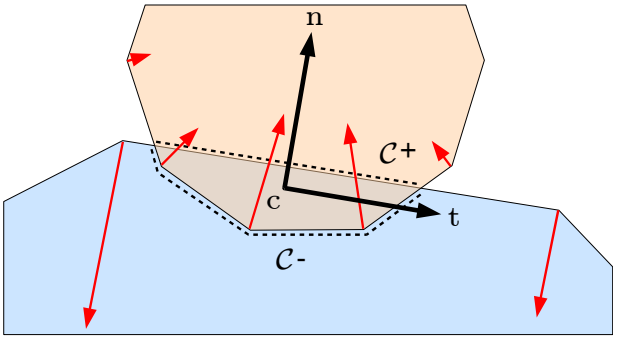


Figure 4: *Two intersecting objects and the contact model. The gradient of the intersection volume appears as red lines at each vertex. The contact frame used for friction is built based on the estimated normal direction* $\mathbf{n}$.

4.2 Friction

Friction forces counteract relative velocity in the tangent plane. In the standard contact model, a friction force is applied at the contact point. We need to modify this definition because our contact force is actually a pressure applied to a contact surface. The sum of the repulsion forces applied to the vertices of one object provides us with the approximate direction of the contact normal vector $\mathbf{n}$. We then compute two orthogonal unit vectors $\mathbf{t_1}$ and $\mathbf{t_2}$ spanning the tangent plane using Gram-Schmidt orthogonalization. To define the equivalent of the point-contact relative velocity in our model, we need to compute a weighted difference of the velocities on the top and bottom contact surfaces $\mathcal{C}^+$ and $\mathcal{C}^-$ as defined by the normal direction (see Figure 4). Specifically, the weight of each vertex must take into account its associated contact surface area, which we approximate by the volume gradient with respect to the vertex position projected onto the volume-contact normal. Thus, we define

$$\mathbf{v}_{rel} \equiv \sum_k \mathcal{A}_k \dot{\mathbf{p}}_k, \quad \text{with } \mathcal{A}_k = \mathbf{n} \cdot \frac{\partial \mathcal{V}}{\partial \mathbf{p}_k}, \tag{20}$$

where $\dot{\mathbf{p}}_k$ is the velocity at vertex k, and $\mathcal{A}_k$ is the area associated with vertex k projected to the tangent plane. The sign of $\mathcal{A}_k$ depends on the orientation of the vertex's associated surface with respect to the contact normal $\mathbf{n}$. This ensures that $\mathbf{v}_{rel}$ will be zero when the two intersecting objects have the same velocity. Note that $\mathbf{v}_{rel}$ has units of velocity times area, so the product of $\mathbf{v}_{rel}$ with our

contact pressure (force per unit area) gives power (velocity times force), like in the standard contact model.

Viscous contact forces are straightforward, and can applied proportionally to the velocity:

$$\mathbf{f}_k = -\nu\, \mathbf{v}_{rel}\, \mathcal{A}_k.$$

To enforce sticking in Coulomb friction, we must constrain the projections of the relative velocity to the basis vectors of the tangent plane $\mathbf{t_1}$ and $\mathbf{t_2}$. At the velocity level, a no-slip constraint can be written as

$$\begin{pmatrix} \mathcal{A}_1\mathbf{t_1}^T \cdots \mathcal{A}_n\mathbf{t_1}^T \\ \mathcal{A}_1\mathbf{t_2}^T \cdots \mathcal{A}_n\mathbf{t_2}^T \end{pmatrix} \begin{pmatrix} \Delta\dot{\mathbf{p}}_1 \\ \vdots \\ \Delta\dot{\mathbf{p}}_n \end{pmatrix} = \begin{pmatrix} -\mathbf{t_1}^T\mathbf{v}_{rel} \\ -\mathbf{t_2}^T\mathbf{v}_{rel} \end{pmatrix},$$

where the $\Delta\dot{\mathbf{p}}_k$ are the velocity corrections necessary to cancel the current sliding velocity. For Coulomb friction instead of using this no-slip condition, we can use Equations 8, 9, and 10, where the matrix $\mathbf{T}$ is now constructed by weighting the symmetric basis formed by the contact frame tangent vectors $\mathbf{t_1}$ and $\mathbf{t_2}$,

$$\mathbf{T} = \begin{pmatrix} \mathcal{A}_1\mathbf{t_1}^T & \cdots & \mathcal{A}_n\mathbf{t_1}^T \\ \mathcal{A}_1\mathbf{t_2}^T & \cdots & \mathcal{A}_n\mathbf{t_2}^T \\ -\mathcal{A}_1\mathbf{t_1}^T & \cdots & -\mathcal{A}_n\mathbf{t_1}^T \\ -\mathcal{A}_1\mathbf{t_2}^T & \cdots & -\mathcal{A}_n\mathbf{t_2}^T \end{pmatrix} \frac{\partial \mathbf{p}}{\partial \mathbf{q}}. \tag{21}$$

This will let us compute the appropriate relative velocity for our volume contact, as well as providing a basis for applying equal and opposite friction forces on the DOFs.

Our net friction forces are centered on the barycenter of each collision surface $\mathcal{C}^+$ and $\mathcal{C}^-$. They are parallel with opposite directions but, in contrast with the repulsion forces, they are not aligned and this generates ghost torques. This is due to the non-physical situation where two objects intersect each other. We can compute the contact center, at the origin of the contact frame shown in Figure 4, as the barycenter of the contact surfaces: $\mathbf{c} = \sum_k \mathcal{A}_k\mathbf{p}_k / \sum_k \mathcal{A}_k$. When volumetric velocity fields are available, forces applied to $\mathbf{c}$ can be propagated to the objects using the transpose of the velocity mapping. We can thus apply all the contact forces at this point rather than the surface vertices, obtain perfectly aligned opposite forces and avoid the ghost torque. We have not found this useful in practice, because the offset between the force directions (and thus the intensity of the ghost torque) is proportional to the intersection depth, which is small in our experiments.

4.3 Animation Loop

Let us now consider the case where multiple objects are in the system. We will use the vector $\boldsymbol{\mathcal{V}}(\mathbf{q})$ to represent the interpenetration volumes between each pair of objects, and we set up the MLCP problems following the description in Sections 3.1 and 4.1. Our animation algorithm starts with the velocity MLCP of Equations 16 and 17, and additionally includes the frictional constraints. We then update the velocities and the positions. The velocities do not necessarily produce a position that will meet the $\mathcal{V}(\mathbf{q}_{0+h}) \leq \epsilon_\mathcal{V}$ constraint, thus we perform a post stabilization step. Due to the non-linearity of the non-interpenetration constraint, we perform multiple iterations and monitor the intersection volume using Newton-Raphson's algorithm.

The post step position correction can ensure non-penetration between colliding objects; however, we must maintain a small amount of intersection rather than removing it completely. The volume derivatives used for the contact and friction formulations can lack precision if the interpenetration volume computed from the LDI involves only a few pixels. In practice, for non-breaking contacts we maintain an interpenetration layer that has a thickness of approximately half a pixel (except in the left of Figure 1, for a better view of the intersection volumes). But this measure of the intersection depth is not available in our contact model. Fortunately, we can estimate the surface of the contact by summing the positive $\mathcal{A}_k$ of Equation 20. From this we compute the average intersection depth within a contact, and estimate the necessary volume variation to preserve the average depth within the desired range. Thus, we automatically regulate the value of $\epsilon_\mathcal{V}$ in the position MLCP of Equations 18 and 19 and we perform LDI collision detection to compute the new intersection volume and its Jacobian. This loop is repeated until the desired precision is obtained or a maximum number of iterations is reached. Each position correction deals only with contact volume correction because the friction constraints do not apply.

Algorithm 1 Animation Loop

Input: current state $\mathbf{q}_0, \dot{\mathbf{q}}_0$
Output: next state $\mathbf{q}_{0+h}, \dot{\mathbf{q}}_{0+h}$

```
Δq̇ ← MLCPsolve velocity update with friction
q̇_{0+h} ← q̇_0 + Δq̇
q_{0+h} ← q_0 + h q̇_{0+h}
Compute LDI: intersection V(q_{0+h}) and Jacobian
while V(q_{0+h}) > ε_V do
    Δq ← MLCPsolve position stabilization
    q_{0+h} ← q_{0+h} + Δq
    Compute LDI: intersection V(q_{0+h}) and Jacobian
end while
```

We implement the velocity update in two steps. The first step, prediction, performs an unconstrained implicit integration of the body forces. The second step, correction, modifies the velocities in order to meet the contact constraints. The position stabilization is implemented as a correction of positions. To solve the MLCPs we use a Gauss-Seidel like method (see, *e.g.*, [Duriez et al. 2006]), but a direct solver like Lemke's algorithm could also be used. We have good results with the integration scheme presented here (linearized backward Euler and post stabilization) but other time-stepping schemes could be used. The novelty lies in the use of volume constraints that are computed using LDI collision detection.

5 Multi-Volume Geometric Model

In the previous section, we have shown that the minimization of the intersection volume generates a single scalar equation for the repulsion between two objects. This is independent of their geometric resolution, which results in dramatic simplifications with respect to traditional distance-based methods. A limitation of this approach is that the contact equation holds for the whole contact area, resulting in a single behavior. For Coulomb friction, the whole contact is either sticking, or sliding, or the patch is breaking contact as a whole. We often expect more detailed behaviors. An example of this is shown in Figure 7, where a part of the snake is sliding while the other is sticking. The solution is to split the intersection volume in several parts with independent contact equations. We call this the *multi-volume* model, in contrast to the mono-volume model presented in the previous section.

We divide the intersection volume using a regular grid aligned with the LDI directions, as shown in the second image of Figure 1. This makes it easy to tune the spatial resolution of the contact. Figure 5 represents a cell of the grid. To set up separate contact equations, we need to compute the volume gradient of each cell. In the viewing direction, the boundary of the intersection volume in a cell may be

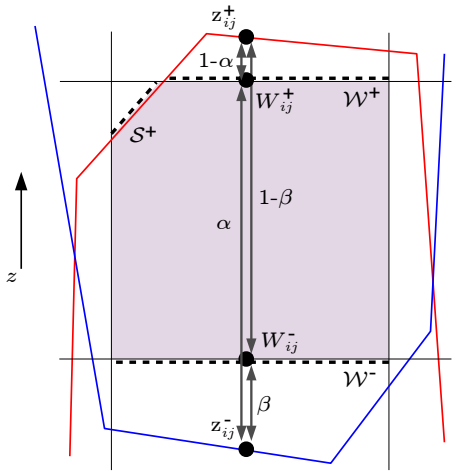


Figure 5: *A slice of the intersection volume in a grid cell.*

composed of up to four surfaces: an upper object surface $\mathcal{S}^+$, an upper cell wall $\mathcal{W}^+$, a lower object surface $\mathcal{S}^-$and a lower cell wall $\mathcal{W}^-$.

The contributions of the object surface pixels to the gradient are given in Section 3.2. The remaining question is how to take into account the pixels of the cell walls. If the cell walls are considered fixed, then their pixels do not contribute to the volume gradient because their depth is independent of the object vertices. For example, the cell highlighted in Figure 5 would only apply a force to the red object, because it does not include the surface of the blue object. Computing the pressures independently in each cell would thus violate Newton's law of equal and opposite forces between the interacting objects. Physical soundness requires the pressure force applied to the cell walls to be dispatched on the object surfaces.

For each pixel of the cell wall, we compute its barycentric coefficient inside the intersection volume in the viewing direction. We do this such that the depth of a pixel of a top cell wall $\mathcal{W}^+$is given by

$$W^+_{ij} = \alpha_{ij} z^+_{ij} + (1 - \alpha_{ij}) z^-_{ij}, \tag{22}$$

while the depth of a pixel of a bottom cell wall $\mathcal{W}^-$is given by

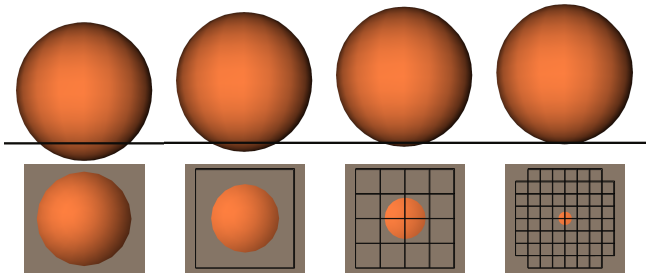
$$W^-_{ij} = \beta_{ij} z^+_{ij} + (1 - \beta_{ij}) z^-_{ij}. \tag{23}$$

An example of these barycentric coefficients is shown in Figure 5. In each cell, the gradient of the intersection volume is thus

$$\begin{aligned} \frac{\partial \mathcal{V}}{\partial \mathbf{p}^z_k} \;=\; & a \sum_{(i,j)\in\mathcal{S}^+_z} \frac{\partial z^+_{ij}}{\partial \mathbf{p}^z_k} - a \sum_{(i,j)\in\mathcal{S}^-_z} \frac{\partial z^-_{ij}}{\partial \mathbf{p}^z_k} \\ & + a \sum_{(i,j)\in\mathcal{W}^+_z} \alpha_{ij} \frac{\partial z^+_{ij}}{\partial \mathbf{p}^z_k} + (1-\alpha_{ij}) \frac{\partial z^-_{ij}}{\partial \mathbf{p}^z_k} \\ & - a \sum_{(i,j)\in\mathcal{W}^-_z} \beta_{ij} \frac{\partial z^+_{ij}}{\partial \mathbf{p}^z_k} + (1-\beta_{ij}) \frac{\partial z^-_{ij}}{\partial \mathbf{p}^z_k} \end{aligned} \tag{24}$$

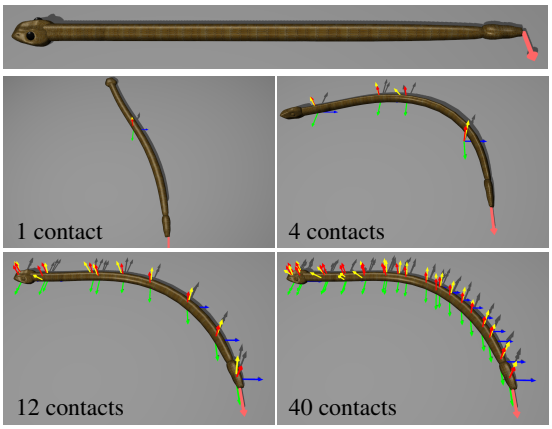
Once the gradient is computed, the contact equations of the cell are straightforward and can be set up the same way as presented in Section 4.

Varying friction behaviors across the contact can greatly enhance the simulation of large deformable objects, as illustrated in the accompanying video. Moreover, separation can also be improved, as



(a) *Initial State* **(b)** *Mono-Volume* **(c)** *Coarse Grid* **(d)** *Fine Grid*

Figure 6: *Intersection after one LCP solution of volume constraint stabilization for grids of different resolution. Increasing the multi-volume precision provides a more accurate linearization of the intersection volume gradients.*

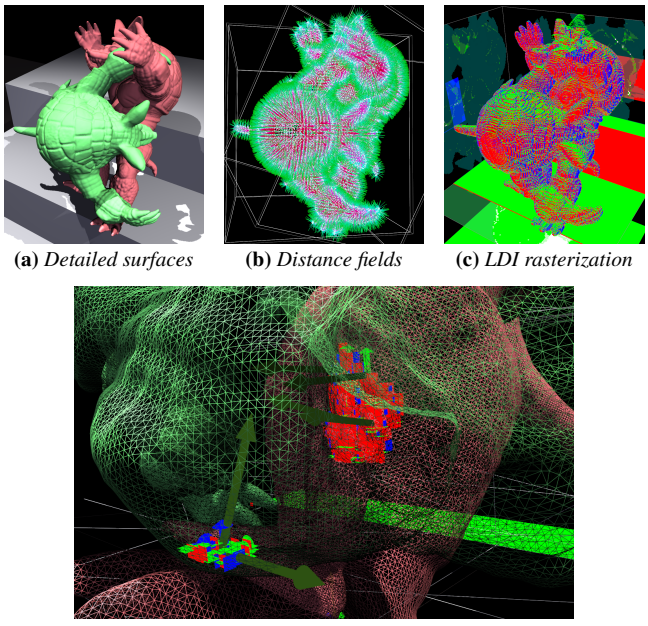


Cell Size (pixels)	Contacts	Avg. Error	Max Error
–	1	26.21 %	61.30 %
250	4	3.47 %	9.94 %
100	7	3.85 %	6.58 %
90	12	0.40 %	0.83 %
50	20	0.25 %	0.46 %
22	40	0.18 %	0.35 %

Figure 7: *Multi-volume friction. Red, green, and blue arrows show the contact frames, while the yellow arrows denote forces. The table shows the measured difference (as fractions of the length of the snake) between the classical penetration-based approach and multi-volume constraints with varying cell sizes.*

illustrated in Figure 6. The actual volume variation between convex objects is smaller than anticipated using the gradient, due to the fact that the area of the contact surface becomes smaller as the objects move apart, as can be seen in Figure 6. As such, some intersection will remain after the linear solve. A complete interpenetration position correction with the mono-volume approach requires several iterations to converge. Using the multi-volume approach, the most exterior contacts are progressively discarded during the MLCP solution, and the final contact occurs only where the intersection was initially the deepest.

We validated the multi-volume model on the example of a snake pulled laterally illustrated in Figure 7. In the mono-volume model, the snake spins around the contact center located approximately in the middle of its body, while it gently evolves to a nice stable bent shape using the multi-volume model. Additionally, when an object is cut as shown in Figure 2c, the multi-volume model allows the independent processing of each of the parts.



(a) *Detailed surfaces* **(b)** *Distance fields* **(c)** *LDI rasterization*

(d) *Contact volumes on the right arm of the green Armadillo*

Figure 8: *Comparison with distance fields. Our method is able to handle highly detailed meshes, as visible in (d), without any precomputation.*

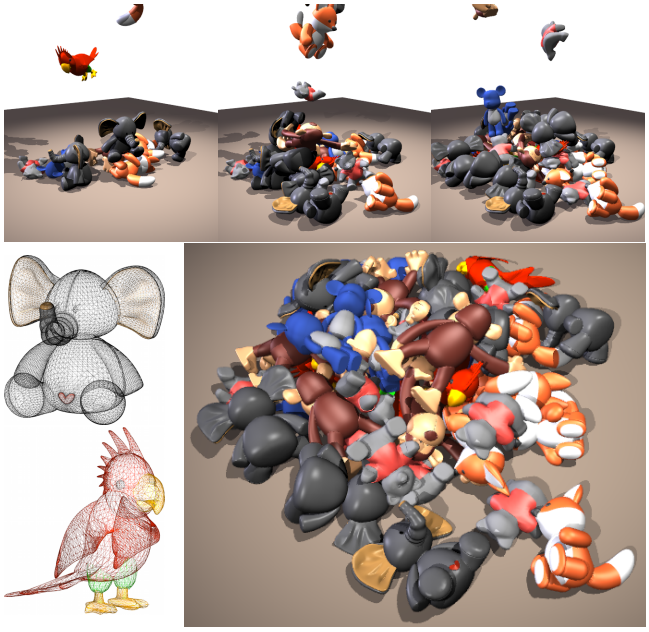


Figure 9: *Stacking of 40 soft toys with complex geometry such as thin ears, sharp features, and a mixture of small and large triangles.*

6 Results

We have successfully applied our method to a variety of scenes including complex rigid and deformable objects. The animation algorithm presented in Section 4.3 is used in all cases. For deformable objects, we use a co-rotational Finite Element Model [Nesme et al. 2005]. We implemented our method within the open-source SOFA library [Allard et al. 2007]. When possible, we compare it with our implementations of distance-based contact models. We performed the simulations on a Intel Core i7 975 CPU with a Nvidia GeForce GTX 285 GPU. Table 1 presents the complexity and measured performance for several example simulations. The *Collision Detection* and *Contact Modeling* columns represent the time taken to compute the LDI and evaluate the intersection volumes and gradients (steps 4 and 8 in Algorithm 1). The *Mechanical Motion* column represents the unconstrained predictive motion due to internal mechanical forces, while the *Constraint Solver* column represents the velocity and position corrections.

The two rigid armadillos in Figure 8 are made of 345 944 triangles each and simulated at 41 fps (not including rendering), with 7 contacts on average, and at most 16 contacts. We also experimented with precomputed distance fields on this scene, since they are a popular and efficient tool for collision detection and response between rigid bodies. The distance fields detect and model collisions faster than our method, allowing refresh rates as high as 57 fps. However, they generate considerably more equations (typically 200 contacts in this example, hence 600 equations including friction), which results in longer solve times. This is especially true when a high resolution distance field is required to handle small geometric details. Our method does not suffer from this problem of mesh resolution sensitivity.

The advantages of our method are clear when distance fields are not available, for instance, when simulating deformable objects or topology changes (Figure 2c). A interesting feature of LDI-based methods is that self-intersections are no more costly to compute than inter-objects collisions. This is illustrated in Figure 2a, where 3 octopuses are simulated at interactive rates, while touching along multiple and complex contact and self-contact areas. The hand grasping the deformable object in Figure 2b is simulated at 54 fps, with 10 contacts, while the proximity-based method runs only at 23 fps due to the complexity of the collision detection and the larger number of contacts, up to 70 in this example.

The complex medical scene in Figure 1 is based on real data. Due to unavoidable data noise and approximations in the reconstruction process, the organs intersect each other at initialization time, and the distance-based method fails to repell them, making the simulation intractable. In contrast, our volume-based method is robust to deep intersections and has no problems separating the organs on the first time step. No spurious velocity is introduced for this correction since the position correction step does not modify the velocities. This simulation is difficult because the grasping relies on high pressure and friction forces. As expected, the tool eventually slips and releases the organ when pulled too far. This is an excellent example of the stability of static friction in our method; the contact forces stay within the friction cone during the first part of the movement, and then lie on the edge of the cone as the organ slips from the grasp of the tool. Note that in this case the geometry of the tool is thin with very fine teeth. It can be handled by our method but requires very small pixels, which is very costly as the LDI is currently rasterized with an uniform precision over the full scene. While the method could be extended to locally adapt the resolution, which is an interesting avenue for future work, we instead replaced the collision mesh for this tool with a slightly simpler and thicker version. The scene includes 550 000 triangles, 9 500 degrees of freedom, and runs at 7 fps including rendering.

An obvious limitation of our approach is that interacting objects must have a volume, so very thin objects such as cloth are not supported. Likewise, since our LDI computations provide intersections at discrete time steps, it is also possible for small fast moving objects to tunnel through one another. For both of these cases, methods based on continuous collision detection are more suitable.

Simulation		Objects	Triangles	LDI		Contacts	Collision Detection	Contact Modeling	Mechanical Motion	Constraint Solver	Iterations Per Second
				Pixels	Layers						
Medical (Figure 1)		9	550K	224×168	32	16	22.69 ms	1.35 ms	17.85 ms	32.45 ms	13.3
Octopuses (Figure 2a)		3	46K	392×352	28	110	5.67 ms	1.20 ms	43.43 ms	12.34 ms	15.6
Grasping (Figure 2b)		2	9K	112×112	14	10	2.73 ms	0.97 ms	7.36 ms	6.18 ms	54.0
Snake (Figure 7)		1	7K	560×336	10	1	3.45 ms	1.01 ms	2.72 ms	1.53 ms	109
						6	3.41 ms	1.00 ms	2.87 ms	1.86 ms	104
						18	3.45 ms	0.95 ms	2.98 ms	3.09 ms	91.9
						41	3.46 ms	0.84 ms	2.99 ms	7.62 ms	65.3
Armadillo (Figure 8)	Rigids	2	*128×107×98 distance field*			255	10.46 ms	0.16 ms	0.38 ms	6.68 ms	56.7
			692K	280×240	22	16	18.48 ms	1.15 ms	0.48 ms	4.08 ms	41.1
	FEM	2	692K	208×192	26	28	18.47 ms	1.53 ms	13.54 ms	17.98 ms	19.3
Soft Toys (Figure 9)		10	205K	456×456	24	99	16.61 ms	1.64 ms	71.24 ms	34.82 ms	7.29
		20	379K	1232×472	34	242	28.79 ms	2.37 ms	146.94 ms	207.02 ms	2.44
		40	725K	1232×528	46	544	44.45 ms	3.43 ms	309.48 ms	723.83 ms	0.88

Table 1: *Complexity and measured performance for example simulations. The Objects, Triangles, LDI and Contacts columns respectively present the number of simulated objects, triangles, and upper limits on the LDI dimensions and generated contacts. The rest of the table presents the average time spent in the four main computation steps, as well as the overall speed of the simulation (excluding rendering).*

Note that we need to maintain a small intersection to model the velocity constraints necessary for resting contacts with dry friction. For each contact volume, we are able to do this by controlling the average intersection depth. This average depth could be much smaller than the true maximum interpenetration depth, but we have not observed any problems related to this in practice.

7 Conclusion

We have presented a new formulation for contact using a hard constraint on the intersection volume. As such, the number of equations is small and independent of geometric complexity. We have extended the formulation to include Coulomb friction, and introduced a multi-volume grid model to better accommodate complex behavior across multiple contact patches. Our method is robust to deep interpenetrations and allows us to simulate frictional contact between complex deformable bodies at unprecedented rates.

Our results show examples that demonstrate that our method performs well for simulations involving sliding, rolling, and resting contact. In general, our multi-volume constraints can be used for a variety of difficult simulation scenarios involving both rigid and deformable objects in frictional contact, such as surgery simulation, stacking, grasping and cutting.

References

ALLARD, J., COTIN, S., FAURE, F., BENSOUSSAN, P.-J., POYER, F., DURIEZ, C., DELINGETTE, H., AND GRISONI, L. 2007. SOFA – an open source framework for medical simulation. In *Medicine Meets Virtual Reality (MMVR'15)*, 13–18. http://www.sofa-framework.org.

ANITESCU, M., AND HART, G. D. 2004. Constraint-stabilized time-stepping approach for rigid multibody dynamics with joints, contact and friction. *International Journal for Numerical Methods in Engineering 60*, 14, 2335–2371.

ANITESCU, M., AND POTRA, F. 1997. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics 14*, 3, 231–247.

ASCHER, U. M., AND PETZOLD, L. R. 1998. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

BACIU, G., AND WONG, W. S.-K. 2004. Image-based collision detection for deformable cloth models. *IEEE Transactions on Visualization and Computer Graphics 10*, 6, 649–663.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 98*, ACM, 43–54.

BARAFF, D. 1991. Coping with friction for non-penetrating rigid body simulation. *Computer Graphics (Proceedings of SIGGRAPH 91) 25*, 4, 31–41.

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

BARBIČ, J., AND JAMES, D. 2007. Time-critical distributed contact for 6-DoF haptic rendering of adaptively sampled reduced deformable models. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 171–180.

BATTY, C., BERTAILS, F., AND BRIDSON, R. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Transactions on Graphics 26*, 3, 100.

BAUMGARTE, J. 1972. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering 1*, 1–16.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of SIGGRAPH 2002*, ACM, 594–603.

CLINE, M. B., AND PAI, D. K. 2003. Post-stabilization for rigid body simulation with contact and constraints. In *IEEE International Conference on Robotics and Automation*, 3744–3751.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. Dynamic real-time deformations using space and time adaptive sampling. In *Proceedings of SIGGRAPH 2001*, ACM, 31–36.

DURIEZ, C., DUBOIS, F., KHEDDAR, A., AND ANDRIOT, C. 2006. Realistic haptic rendering of interacting deformable objects in virtual environments. *IEEE Transactions on Visualization and Computer Graphics 12*, 1, 36–47.

ERLEBEN, K. 2007. Velocity-based shock propagation for multibody dynamics animation. *ACM Transactions on Graphics 26*, 2, 12.

FAURE, F., BARBIER, S., ALLARD, J., AND FALIPOU, F. 2008. Image-based collision detection and response between arbitrary volume objects. In *SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 155–162.

GOTTSCHALK, S., LIN, M. C., AND MANOCHA, D. 1996. OBBTree: a hierarchical structure for rapid interference detection. In *Proceedings of SIGGRAPH 96*, ACM, 171–180.

GUENDELMAN, E., BRIDSON, R., AND FEDKIW, R. Nonconvex rigid bodies with stacking. *ACM Transactions on Graphics 22*, 3, 871–878.

HARMON, D., VOUGA, E., TAMSTORF, R., AND GRINSPUN, E. 2008. Robust treatment of simultaneous collisions. *ACM Transactions on Graphics 27*, 3, 1–4.

HARMON, D., VOUGA, E., SMITH, B., TAMSTORF, R., AND GRINSPUN, E. 2009. Asynchronous contact mechanics. *ACM Transactions on Graphics 28*, 3.

HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2003. Real-time volumetric intersections of deforming objects. In *Proceedings of Vision, Modeling, Visualization (VMV)*, 461–468.

HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2004. Detection of collisions and self-collisions using image-space techniques. In *Proceedings of WSCG'04*, 145–152.

HUBBARD, P. M. 1995. *Collision Detection for Interactive Graphics Applications*. PhD thesis, Brown University.

JAMES, D. L., AND PAI, D. K. 2004. BD-tree: output-sensitive collision detection for reduced deformable models. *ACM Transactions on Graphics 23*, 3, 393–398.

KAUFMAN, D. M., EDMUNDS, T., AND PAI, D. K. 2005. Fast frictional dynamics for rigid bodies. *ACM Transactions on Graphics 24*, 3, 946–956.

KAUFMAN, D. M., SUEDA, S., JAMES, D. L., AND PAI, D. K. 2008. Staggered projections for frictional contact in multibody systems. *ACM Transactions on Graphics 27*, 5, 1–11.

LLOYD, J. E. 2005. Fast implementation of Lemke's algorithm for rigid body contact simulation. In *IEEE International Conference on Robotics and Automation*, 4538–4543.

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

MÜLLER, M. 2008. Hierarchical position based dynamics. In *VRIPHYS 08: Fifth Workshop in Virtual Reality Interactions and Physical Simulations*, Eurographics Association, 1–10.

NEALEN, A., MÜLLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. 2005. Physically based deformable models in computer graphics. In *Eurographics 2005 - State of the Art Reports*, 71–94.

NESME, M., PAYAN, Y., AND FAURE, F. 2005. Efficient, physically plausible finite elements. In *Eurographics 2005 - Short Papers*, 77–80.

OTADUY, M. A., JAIN, N., SUD, A., AND LIN, M. C. 2004. Haptic display of interaction between textured models. In *Proceedings of IEEE Visualization Conference*, 297–304.

OTADUY, M. A., GERMANN, D., REDON, S., AND GROSS, M. 2007. Adaptive deformations with fast tight bounds. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 181–190.

OTADUY, M. A., TAMSTORF, R., STEINEMANN, D., AND GROSS, M. 2009. Implicit contact handling for deformable objects. *Computer Graphics Forum (Proceedings of Eurographics) 28*, 2, 559–568.

PABST, S., THOMASZEWSKI, B., AND STRASSER, W. 2009. Anisotropic friction for deformable surfaces and solids. In *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 149–154.

PARKER, E. G., AND O'BRIEN, J. F. 2009. Real-time deformation and fracture in a game environment. In *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 165–175.

PAULY, M., PAI, D. K., AND GUIBAS, L. J. 2004. Quasi-rigid objects in contact. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 109–119.

PROVOT, X. 1997. Collision and self-collision handling in cloth model dedicated to design garments. In *Proceedings of 8th Eurographics Workshop on Animation and Simulation*, 177–189.

SHI, L., YU, Y., BELL, N., AND FENG, W.-W. 2006. A fast multigrid algorithm for mesh deformation. *ACM Transactions on Graphics 25*, 3, 1108–1117.

STEWART, D. E., AND TRINKLE, J. C. 1996. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal of Numerical Methods Engineering 39*, 15, 2673–2691.

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

SUD, A., GOVINDARAJU, N., GAYLE, R., KABUL, I., AND MANOCHA, D. 2006. Fast proximity computation among deformable models using discrete voronoi diagrams. *ACM Transactions on Graphics 25*, 3, 1144–1153.

TESCHNER, M., KIMMERLE, S., HEIDELBERGE, B., ZACHMANN, G., RAGHUPATHI, L., FUHRMANN, A., CANI, M.-P., FAURE, F., MAGNENAT-THALMANN, N., STRASSER, W., AND VOLINO, P. 2004. Collision detection for deformable objects. In *Eurographics 2004 - State of the Art Reports*.

VAN DEN BERGEN, G. 1997. Efficient collision detection of complex deformable models using aabb trees. *Journal of Graphics Tools 2*, 4, 1–13.

VASSILEV, T., SPANLANG, B., AND CHRYSANTHOU, Y. 2001. Fast cloth animation on walking avatars. *Computer Graphics Forum (Proceedings of Eurographics) 20*, 3, 260–267.

VOLINO, P., AND MAGNENAT-THALMANN, N. 1995. Collision and self-collision detection: Efficient and robust solutions for higly deformable surfaces. In *Computer Animation and Simulation '95*, 55–65.

WONG, W. S.-K., AND BACIU, G. 2005. GPU-based intrinsic collision detection for deformable surfaces: Collision detection and deformable objects. *Computer Animation and Virtual Worlds 16*, 3-4, 153–161.