Implicit Multibody Penalty-based Distributed Contact

Hongyi Xu, *Member, IEEE,* Yili Zhao, *Member, IEEE,* and Jernej Barbič, *Member, IEEE*

Abstract—The penalty method is a simple and popular approach to resolving contact in computer graphics and robotics. Penalty-based contact, however, suffers from stability problems due to the highly variable and unpredictable net stiffness, and this is particularly pronounced in simulations with time-varying distributed geometrically complex contact. We employ semi-implicit integration, exact analytical contact gradients, symbolic Gaussian elimination and a SVD solver to simulate stable penalty-based frictional contact with large, time-varying contact areas, involving many rigid objects and articulated rigid objects in complex conforming contact and self-contact. We also derive implicit proportional-derivative control forces for real-time control of articulated structures with loops. We present challenging contact scenarios such as screwing a hexbolt into a hole, bowls stacked in perfectly conforming configurations, and manipulating many objects using actively controlled articulated mechanisms in real time.

Index Terms—Computer Graphics, Physically based modeling, Animation, Kinematics and dynamics

1 Introduction

Simulating contact between geometrically complex objects is one of the key tasks in computer animation. Penalty-based methods are perhaps the oldest method to provide contact response. The basic idea of penalty methods is straightforward: as an object enters collision, one adds a penalty spring force that pushes the object out of collision. The method is conceptually simple, easy to implement, and the forces can be evaluated quickly. It works reasonably well with simple contact scenarios, such as a single (near-)planar contact, or collisions involving a small number of vertices. With large examples and complex contact, however, penalty methods are less reliable. Unless the timestep is made very small, they are often prone to numerical instabilities due to the high and unpredictable stiffness variations encountered at any timestep.

Recent trends in rigid body simulation have focused on delivering larger and larger simulations at increasingly faster speeds [1]. In this paper, we show how *geometrically complex* penalty-based distributed contact between many rigid objects and articulated objects can be stably simulated, by employing exact analytical contact gradients, symbolic Gaussian elimination, SVD solver, and semi-implicit integration. *Distributed* contact is the most general type of contact where contact areas may be many and of varying

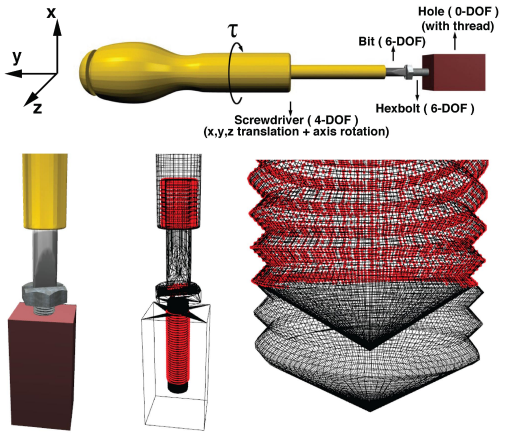


Fig. 1: Stable time-varying penalty distributed contact between several objects: *User applies a torque to the screwdriver to screw the hexbolt into a hole. This 4-body simulation has large, time-varying contact areas: 30,000 contacts (in red) at each timestep on average.*

size in both space and time. With complex distributed contact, penalty forces are temporally incoherent. The net stiffness felt by the object varies abruptly as vertices enter in and out of contact, and this is especially pronounced in conforming contact configurations (Figure 1). Employing contact gradients and implicit integration makes it possible to dissipate stiff contact, much like the stability of elasticity solvers improves under implicit integration. Given the contact gradient of each contact, our approach assembles a global linear system of equations for the change in linear and angular momenta of each object, where each pair of colliding objects contributes four 6×6 matrix blocks.

Our method improves simulation stability regardless of the number of contacts and the sizes and orientations of contact areas. It applies to all penalty-based methods, regardless of the particular definition of the penalty force, such as volume penetration, or penetration depth, as long

- *All authors are with the Department of Computer Science, University of Southern California, Los Angeles, CA, 90089. E-mail: hongyixu@usc.edu, yilizhao@usc.edu, jnb@usc.edu.*
- *Hongyi Xu and Yili Zhao both contributed equally to this paper, and should be considered equal first authors.*

- *Digital Object Identifier no. 10.1109/TVCG.2014.2312013*

Fig. 2: Actively controlled articulated excavator in distributed contact: *Left: articulated structure with three PD-servo controlled prismatic joints. Right: user controls the excavator to pick up and move rocks in real-time (9 rocks; 50 fps).*

as forces and force gradients can be computed. The method handles both transient and continuous contact, and supports friction. It supports complex, conforming, contact between multiple rigid objects (Figure 1), as well as articulated rigid objects in contact and self-contact. Our constraints are modeled using maximal coordinates and permit arbitrary looping (Figure 2). We also demonstrate how to add active control to articulated rigid object in contact, using implicit proportional-derivative control forces. We limit ourselves to penalty-based methods in this paper, and compare our work to a constraint-based method (Bullet Physics [2]) in Results (Section 6). Our technical contributions were designed to accommodate complex, distributed penalty contact and self-contact, and include:

- implicit global system for multiple rigid and articulated objects, with 6×6 blocks for each rigid body,
- exact analytical contact gradient derivation for contact forces with varying normals,
- constraint gradient and Hessian for simulations with constraints and complex contact, and
- SVD solver for singular overconstrained linear systems arising with articulated bodies with loops.

2 Related Work

Contact simulation methods in computer graphics can be loosely categorized into constraint-based methods [3], [4], [5], [6], [7], impulse methods [8], and penalty-based methods [9]. The first two classes of methods generally produce more plausible simulations at the cost of more computation, typically scaling superlinearly in the number of contacts, with some exceptions [10]. In this paper, we limit ourselves to penalty methods. Our method runs in time linear in the number of contacts, and scales to large contact areas and distributed contact.

Penalty methods are commonly used in computer graphics, especially in applications where accuracy is less important than speed [11]. Like our method, many methods use distance fields to compute the penetration depth for penalty forces [12], [13], [11]. A frictional model for penalty-based simulation was proposed in [14], and we adopt their frictional model in our work. Many authors improved the definition of penetration depth and temporal stability of penalty forces [15], [16], [17]. Stiff springs can be replaced for potential barriers [18], or integrated at a higher rate than the rest of the system [19]. Recently, Tang and colleagues [20] presented a method that combines continuous collision detection with *continuous* penalty force response, using explicit integration. Our implicit method is complementary and could be combined with their method, as it demonstrates how to make penalty-based simulations more stable in stiff contact simulations (comparison in §6).

Rigid and articulated object simulation is a key concept in computer animation [21], and we refer the reader to a good recent survey [1]. State-of-the-art industry tools exist [2] that can simulate thousands of rigid objects at interactive rates using constraint-based approaches, albeit in most cases with relatively simple geometry and a low number of contact points per object. Symplectic integrators can preserve rigid object invariants, such as angular momentum or energy [22], [23]. Articulated rigid objects can be simulated in reduced coordinates [24] or using maximal coordinates, which can be done with Lagrange multipliers [25], or impulses [26]. Our method employs Lagrange multipliers and Baumgarte stabilization [27], but could be extended to support impulses as well.

Implicit methods are commonly employed for numerical integration of stiff equations, including contact [28], although the majority of applications have been limited to contact between two objects. Implicit integration was used to simulate stable penalty-based contact between two rigid objects [29], and applied to haptic rendering [30]. We accelerate their method using symbolic Gaussian elimination, and extend it to multiple objects and articulated objects, including self-contact. Otaduy and colleagues [31] gave an implicit constraint-based method for deformable object contact, and Hernandez and colleagues [32] used an implicit formulation for stable simulation of articulated structures without loops. We support articulated structures with loops in complex, distributed contact, undergoing active control. Complex, penalty-based distributed contact between *two* rigid objects has been simulated at haptic rates in the Voxmap-Pointshell System (VPS) [33], [34]. Barbič and James [35] extended the method to deformable objects and improved contact stability by using distance fields. Similarly, our work uses points and distance fields, but supports contact between many rigid and articulated objects.

Image-based volume contact approaches [36], [37], [38] ease the computation of repulsion forces by minimizing contact volumes. These methods rely on GPU computation and have incorporated penalty-like contact forces, including approximate contact force gradients and implicit integration [36]. Because our paper is generic with respect to the

penalty-based force implementation, the contact forces and gradients provided by image-based volume methods could be used as input to our method. We focus on CPU-based computation in this paper, and compute exact contact force and torque gradients in our specific penalty implementation.

3 Implicit Rigid Contact

In our method, the state of a rigid body is represented as: $S(t) = [x(t), q(t), P(t), L(t)] \in \mathbb{R}^{13}$, where x is the position of the center of mass, q is a quaternion describing the orientation, and P and L are linear and angular momentum, respectively. The equation of motion for a single rigid body can be expressed as the following ODEs [39], [30]

$$\dot{\mathbf{S}}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{q}(t) \\ \dot{P}(t) \\ \dot{L}(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{m}P(t) \\ \frac{1}{2}\hat{\omega}q(t) \\ F(t) \\ \tau(t) \end{bmatrix}, \tag{1}$$

where m is the mass of the body, $\hat{\omega}$ is a quaternion whose scalar part is 0 and vector part is the angular velocity ω, and F and τ denote the external force and torque. Note that the angular velocity ω can be computed as $\omega = R\bar{J}^{-1}R^T L$, where R is the current rotation matrix and $\bar{J}$ is the inertia tensor around the center of mass, computed in the rest configuration. Let h denote the time step. Using the semi-implicit backward Euler scheme, Equation 1 can be discretized with first-order precision as:

$$S(t+h) = S(t) + h\dot{S}(t+h), \tag{2}$$

$$\left(I - h\frac{\partial \dot{S}}{\partial S}\right)\left(S(t+h) - S(t)\right) = h\dot{S}(t). \tag{3}$$

The system matrix is [30]

$$I - h\frac{\partial \dot{S}}{\partial S} = \begin{bmatrix} I & 0 & -\frac{h}{m}I & 0 \\ 0 & I - h\frac{\partial \dot{q}}{\partial q} & 0 & -h\frac{\partial \dot{q}}{\partial L} \\ -h\frac{\partial F}{\partial x} & -h\frac{\partial F}{\partial q} & I - h\frac{\partial F}{\partial P} & -h\frac{\partial F}{\partial L} \\ -h\frac{\partial \tau}{\partial x} & -h\frac{\partial \tau}{\partial q} & -h\frac{\partial \tau}{\partial P} & I - h\frac{\partial \tau}{\partial L} \end{bmatrix}. \tag{4}$$

At this point, we depart from [39] and [30], who solve the 13×13 dense non-symmetric system given by (4). Instead, we exploit the identity and zero entries in the system matrix (4) and first perform block Gaussian elimination (using algebra; not numerically), to arrive at a 6×6 system, which can be solved more quickly, especially when simulating multiple rigid objects in contact in later sections. We measured a 2.6x overall speedup (including system preparation time) for a single rigid object in contact, compared to solving the 13x13 system directly. The 6×6 system is (derivation is in Section 3.1)

$$G\begin{bmatrix} \Delta P \\ \Delta L \end{bmatrix} = h\begin{bmatrix} F + h\left(\frac{\partial F}{\partial x}v + \frac{\partial F}{\partial \theta}V\omega\right) \\ \tau + h\left(\frac{\partial \tau}{\partial x}v + \frac{\partial \tau}{\partial \theta}V\omega\right) \end{bmatrix}, \tag{5}$$

$$\text{for } G = \begin{bmatrix} I - \frac{h}{m}\left(\frac{\partial F}{\partial v} + h\frac{\partial F}{\partial x}\right) & -h\left(\frac{\partial F}{\partial \omega} + h\frac{\partial F}{\partial \theta}V\right)J^{-1} \\ -\frac{h}{m}\left(\frac{\partial \tau}{\partial v} + h\frac{\partial \tau}{\partial x}\right) & I - h\left(\frac{\partial \tau}{\partial \omega} + h\frac{\partial \tau}{\partial \theta}V\right)J^{-1} \end{bmatrix}, \tag{6}$$

where $Q \in \mathbb{R}^{4\times 3}$ is the matrix with the property that $Qx = \frac{1}{2}\hat{x}q$, for quaternion q and $\hat{x}$, $V = 4Q^T(I - h\frac{\partial \dot{q}}{\partial q})^{-1}Q$, and $\frac{\partial \dot{q}}{\partial q}$ is given in Section 3.1. Quantity $\partial\theta$ denotes an infinitesimal angular displacement. All quantities are evaluated at time t. Note that such a system can be used for implicit integration of a single rigid object in arbitrary, distributed contact with a fixed environment (Figure 3).

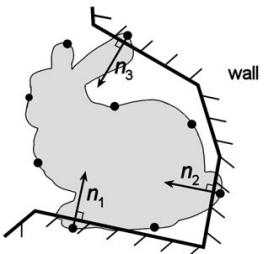


Fig. 3: *Distributed contact*

3.1 Derivation of the 6×6 linear system (Eq. 5)

By inserting (4) into (3), we obtain

$$A\begin{bmatrix} \Delta x \\ \Delta q \\ \Delta P \\ \Delta L \end{bmatrix} = h\begin{bmatrix} \frac{P_t}{m} \\ \frac{1}{2}\hat{\omega}_t q_t \\ F_t \\ \tau_t \end{bmatrix}, \quad \text{where} \tag{7}$$

$$A = \begin{bmatrix} I & 0 & -\frac{h}{m}I & 0 \\ 0 & I - h\frac{\partial \dot{q}}{\partial q} & 0 & -hQ_t J_t^{-1} \\ -h\frac{\partial F}{\partial x} & -h\frac{\partial F}{\partial q} & I - h\frac{\partial F}{\partial P} & -h\frac{\partial F}{\partial L} \\ -h\frac{\partial \tau}{\partial x} & -h\frac{\partial \tau}{\partial q} & -h\frac{\partial \tau}{\partial P} & I - h\frac{\partial \tau}{\partial L} \end{bmatrix}. \tag{8}$$

We then use diagonal blocks I and $I - \frac{\partial \dot{q}}{\partial q}$ to eliminate the blocks $-h\partial F/\partial x$, $-h\partial F/\partial q$, $-h\partial\tau/\partial x$, $-h\partial\tau/\partial q$ (block Gaussian elimination), converting (7) into

$$A'\begin{bmatrix} \Delta x \\ \Delta q \\ \Delta P \\ \Delta L \end{bmatrix} = h\begin{bmatrix} \frac{P_t}{m} \\ \frac{1}{2}\hat{\omega}_t q_t \\ F_t + h\frac{\partial F}{\partial x}v_t + h\frac{\partial F}{\partial q}Q_t\omega_t \\ \tau_t + h\frac{\partial \tau}{\partial x}v_t + h\frac{\partial \tau}{\partial q}Q_t\omega_t \end{bmatrix}, \tag{9}$$

where

$$A' = \begin{bmatrix} I & 0 & -\frac{h}{m}I & 0 \\ 0 & I & 0 & -hQ_t J_t^{-1} \\ 0 & 0 & I - h\frac{\partial F}{\partial P} - \frac{h^2}{m}\frac{\partial F}{\partial x} & -h\frac{\partial F}{\partial L} - h^2\frac{\partial F}{\partial q}Y_t \\ 0 & 0 & -h\frac{\partial \tau}{\partial P} - \frac{h^2}{m}\frac{\partial \tau}{\partial x} & I - h\frac{\partial \tau}{\partial L} - h^2\frac{\partial \tau}{\partial q}Y_t \end{bmatrix}, \tag{10}$$

for $Y_t = (I - h\frac{\partial \dot{q}}{\partial q})^{-1}Q_t J_t^{-1}$. The $\frac{\partial \dot{q}}{\partial q}$ term is [30]

$$\frac{\partial \dot{q}}{\partial q_i} = \frac{\partial Q}{\partial q_i}\omega + Q\frac{\partial \omega}{\partial q_i} \tag{11}$$

$$\frac{\partial \omega}{\partial q_i} = \left(\frac{\partial R}{\partial q_i}\bar{J}^{-1}R^T + R\bar{J}^{-1}\frac{\partial R}{\partial q_i}^T\right)L. \tag{12}$$

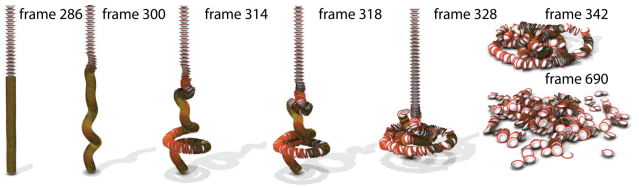


Fig. 4: 512 bowls falling exactly on-center. *Interesting buckling patterns occur due to severe conforming contact.*

Quantity $\frac{\partial R}{\partial q_i}$ is standard, and defined in [29]. Here $\frac{\partial \dot{q}}{\partial q}$ is assembled after $\frac{\partial \dot{q}}{\partial \dot{q}_i}$ is computed for each row q_i of q.

One can easily verify the following equations:

$$\frac{\partial F}{\partial q} = 4\frac{\partial F}{\partial \theta}Q^T, \qquad \frac{\partial \tau}{\partial q} = 4\frac{\partial \tau}{\partial \theta}Q^T,$$
$$\frac{\partial F}{\partial L} = \frac{\partial F}{\partial \omega}J^{-1}, \qquad \frac{\partial \tau}{\partial L} = \frac{\partial \tau}{\partial \omega}J^{-1}. \tag{13}$$

After substituting (13) into (9), the last 6 equations form a 6×6 linear system (Equation 5).

4 Contact between many rigid objects

We now extend our implicit method to contact between many rigid objects (see, for example, Figure 4). We will assemble a global linear system of $6n\times 6n$ equations to handle contact between $n>1$ objects simultaneously. Our work is not specific to a particular penalty-based contact implementation. We assume a penalty implementation which, given the positions and orientations of two rigid bodies, returns the net contact force and torque between them, as well as gradient of force and torque with respect to perturbing either object's position or orientation. We provide such an implementation in Section 4.1.

For $i\neq j$, let $F_{i,j}$ and $\tau_{i,j}$ denote the applied forces and torques from object j to i, respectively, including contact, contact damping and friction forces and torques. The torques on object i are computed with respect to its current center of mass position. Let $F_{i,i}$ and $\tau_{i,i}$ denote the total external force and torque on object i, excluding forces $F_{i,j}$, and including gravity $m_i g$ and any other external forces or torques such as contact with stationary objects (e.g., ground). Let F_i and τ_i denote the total force and torque on object i,

$$F_i = \sum_{k=1}^{n} F_{i,k}, \qquad \tau_i = \sum_{k=1}^{n} \tau_{i,k}. \tag{14}$$

After performing collision detection and evaluating contact gradients, we form and solve the $6n\times 6n$ system for $\Delta S_i = [\Delta P_i, \Delta L_i]^T$,

$$G\begin{bmatrix}\Delta S_1\\ \Delta S_2\\ \vdots\\ \Delta S_n\end{bmatrix} = h\begin{bmatrix}Z_1\\ Z_2\\ \vdots\\ Z_n\end{bmatrix},\ G=\begin{bmatrix}G_{1,1} & G_{1,2} & \cdots & G_{1,n}\\ G_{2,1} & G_{2,2} & \cdots & G_{2,n}\\ \vdots & \vdots & \ddots & \vdots\\ G_{n,1} & G_{n,2} & \cdots & G_{n,n}\end{bmatrix}, \tag{15}$$

where the 6×6 subblocks are obtained as in Section 3:

$$G_{i,j} = \delta_{ij}I - h\begin{bmatrix}\frac{1}{m_j}\left(\frac{\partial F_{i,j}}{\partial v_j}+h\frac{\partial F_{i,j}}{\partial x_j}\right) & \left(\frac{\partial F_{i,j}}{\partial \omega_j}+h\frac{\partial F_{i,j}}{\partial \theta_j}V_{i,j}\right)J_j^{-1}\\ \frac{1}{m_j}\left(\frac{\partial \tau_{i,j}}{\partial v_j}+h\frac{\partial \tau_{i,j}}{\partial x_j}\right) & \left(\frac{\partial \tau_{i,j}}{\partial \omega_j}+h\frac{\partial \tau_{i,j}}{\partial \theta_j}V_{i,j}\right)J_j^{-1}\end{bmatrix}, \tag{16}$$

$$Z_i = \begin{bmatrix}F_i + h\left(\sum_{k=0}^{n}\frac{\partial F_{i,k}}{\partial x_k}v_k + \sum_{k=0}^{n}\left(\frac{\partial F_{i,k}}{\partial \theta_k}V_{i,k}\omega_k\right)\right)\\ \tau_i + h\left(\sum_{k=0}^{n}\frac{\partial \tau_{i,k}}{\partial x_k}v_k + \sum_{k=0}^{n}\left(\frac{\partial \tau_{i,k}}{\partial \theta_k}V_{i,k}\omega_k\right)\right)\end{bmatrix}, \tag{17}$$

$$V_{i,j} = 4Q_j^T\left(I - h\frac{\partial \dot{q}_i}{\partial q_j}\right)^{-1}Q_j, \tag{18}$$

where δ_{ij} is 0 if $i\neq j$, and 1 if $i=j$. Note that for $i\neq j$, we have $V_{i,j}=I$, because $\frac{\partial \dot{q}_i}{\partial q_j}=0$ and $4Q_i^TQ_i=I$. Note that if objects i and j are not in contact, $G_{ij}=G_{ji}=0$. The system (15) is unsymmetric but block-sparse structurally symmetric, and we solve it using the PARDISO solver [40].

Contact islands: At every timestep, we form a graph whose nodes are rigid objects, connected if they are colliding. We then find its connected components (*contact islands*) [39]. The contact islands are decoupled from each other in (15) and are solved independently, which often greatly decreases the system solve time.

Friction is employed in all of our four examples, and helps with stacking. We use the friction model of [14], which models static and dynamic Coulomb friction using anchors and penalty springs, and we briefly restate it here for completeness. When a new contact is detected, an anchor is set at the contact location; a linear spring then pulls the contact point to the anchor in subsequent timesteps. The anchor is released when the contact disappears, or if the static friction force exceeds the normal contact force times the static coefficient of friction, in which case the contact changes to dynamic friction. The magnitude of dynamic friction equals the dynamic coefficient of friction times the normal contact force; its direction is the current velocity, projected to the contact plane and normalized to unit length. If dynamic friction falls below static friction coefficient times the normal force, the anchor is dropped again and contact thereforth uses static friction. Such friction can, for example, correctly model a block sliding or not sliding down an inclined plane based on the relationship between the static coefficient of friction and $\tan(\theta)$, where θ is the inclination angle. Because our system already employs contact damping (both normal and tangential components of velocity), we do not use frictional damping. Because friction stiffness is approximately $10{,}000\times$ smaller than contact stiffness in our examples, we timestep friction forces explicitly; but friction could be treated implicitly and added to Equation 15. The employed friction model is determined locally at each contact, and has the limitation that objects will undergo motion until the friction springs saturate. We observed, however, that the model can keep residual motion small and not easily perceptible. A proper amount of damping helps with such a stabilization.

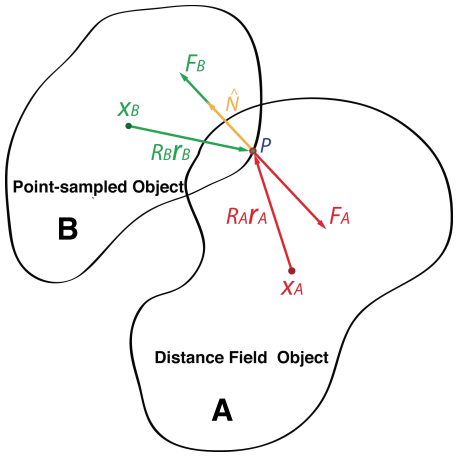


Fig. 5: Penalty contact force and torque *between a point-sampled object and a distance field.*

4.1 Penalty contact force and torque gradients

In this section, we derive the penalty force and torque gradients for a specific penalty-based method of [35]. This model computes penalty forces and torques between a point-sampled object and a distance field object. It avoids distance field gradient discontinuities by using the point's normal as the direction of the penalty force. For shallow contact with sufficient point sampling density, it can be shown that this penalty models gives forces proportional to penetration volume [35]. We use this penalty model everywhere in our experiments.

We extend previous penalty contact gradients [30], [35] that simulated a single object in contact with a static environment, to contact between $n > 1$ dynamic objects. Such contact involves pairs of contacting dynamic objects, and therefore requires a derivation of the force and torque gradients as *each* of the two objects translates and rotates. Because the normals rotate, such an exact gradient is not necessarily symmetric with respect to rotating the first vs. the second object: rotating object A by a small angle θ and keeping B fixed, does not produce the same change in force/torque on A as fixing A and rotating B by $-\theta$. Our work correctly models this effect, whereas [35] simplified the gradient to be symmetric, and approximated the distance field gradient to be oriented in the direction of the contact normal. We do not make any of these approximations. We provide a comparison to [35] in Figure 12. We can also incorporate gradients with respect to velocities and angular velocities (see Section 4.2), useful, for example, with strong damping.

Let us assume that object A is a distance field object, and object B is sampled with points. We will now derive contact forces and gradients between A and B. In our simulations, each object carries both a distance field and points. For every pair of colliding objects, we compute contact forces and gradients twice: with A a distance field object and B point-sampled, and vice-versa. Each object has a local frame, denoted as $\mathcal{F}_A$ and $\mathcal{F}_B$. When collision occurs (Figure 5), the contact forces F_A and F_B, in world coordinate system, are defined as follows [35]

$$F_B = -kd\hat{N}, \qquad F_A = -F_B = kd\hat{N}, \tag{19}$$

$$\tau_B = (R_B r_B) \times F_B, \qquad \tau_A = (R_A r_A) \times F_A, \tag{20}$$

$$P = x_A + R_A r_A = x_B + R_B r_B, \qquad \hat{N} = R_B N, \tag{21}$$

where $d < 0$ is the signed distance field value, k is the stiffness coefficient, $\hat{N} \in \mathbb{R}^3$ is the point's inward normal (w.r.t. object B) in world frame, τ_A, τ_B are contact torques, r_A, r_B are handles in rest configuration expressed in each object's local frame, R_A, R_B are current rotation matrices, x_A and x_B are the current centers of the mass, and N is the point's inward normal in $\mathcal{F}_B$.

We now compute the gradients using an infinitesimal method. Let us assume that objects A and B undergo a (small) displacement $(\Delta x_A, \Delta\theta_A)$ and $(\Delta x_B, \Delta\theta_B)$, respectively. The query point P is on the surface of the pointshell object B, and therefore its position in the frame of reference of B does not change under motion of either A or B. In the frame of reference of A, however, the position of point P changes under motion of A or B because P is attached to B. We have

$$\Delta r_A = R_A^T(\Delta x_B + \Delta\theta_B \times R_B r_B - \Delta x_A - \Delta\theta_A \times R_A r_A), \tag{22}$$

$$\Delta r_B = 0, \qquad \Delta\hat{N} = \Delta\theta_B \times N, \qquad \Delta d = \frac{\partial d}{\partial r_A}\Delta r_A \triangleq g^T \Delta r_A, \tag{23}$$

where $\partial d/\partial r_A$, denoted as g^T, is the gradient of distance field at point P, expressed in $\mathcal{F}_A$. The derivatives of $\hat{N}$, r_A, r_B and d w.r.t. x_A, x_B, θ_A and θ_B are

$$\frac{\partial\hat{N}}{\partial x_A} = 0, \quad \frac{\partial\hat{N}}{\partial\theta_A} = 0, \quad \frac{\partial\hat{N}}{\partial x_B} = 0, \quad \frac{\partial\hat{N}}{\partial\theta_B} = -\tilde{N}, \tag{24}$$

$$\frac{\partial r_A}{\partial x_A} = -R_A^T, \quad \frac{\partial r_A}{\partial x_B} = R_A^T, \quad \frac{\partial r_B}{\partial x_A} = 0, \quad \frac{\partial r_B}{\partial x_B} = 0, \tag{25}$$

$$\frac{\partial r_A}{\partial\theta_A} = R_A^T\widetilde{R_A r_A}, \quad \frac{\partial r_A}{\partial\theta_B} = -R_A^T\widetilde{R_B r_B}, \tag{26}$$

$$\frac{\partial r_B}{\partial\theta_A} = 0, \quad \frac{\partial r_B}{\partial\theta_B} = 0, \tag{27}$$

$$\frac{\partial d}{\partial x_A} = -g^T R_A^T, \quad \frac{\partial d}{\partial x_B} = g^T R_A^T, \tag{28}$$

$$\frac{\partial d}{\partial\theta_A} = g^T R_A^T\widetilde{R_A r_A}, \quad \frac{\partial d}{\partial\theta_B} = -g^T R_A^T\widetilde{R_B r_B}. \tag{29}$$

To compute the gradient of contact force, we apply the product rule,

$$\frac{\partial(d\hat{N})}{\partial x} = d\frac{\partial\hat{N}}{\partial x} + \hat{N} \otimes \left(\frac{\partial d}{\partial x}\right)^T, \tag{30}$$

which gives

$$\frac{\partial F_A}{\partial x_A} = kd\frac{\partial \hat{N}}{\partial x_A} + k\hat{N}\otimes\left(\frac{\partial d}{\partial x_A}\right)^T = -k\hat{N}\otimes(R_A g) \quad (31)$$

$$\frac{\partial F_A}{\partial x_B} = kd\frac{\partial \hat{N}}{\partial x_B} + k\hat{N}\otimes\left(\frac{\partial d}{\partial x_B}\right)^T = k\hat{N}\otimes(R_A g) \quad (32)$$

$$\begin{aligned}\frac{\partial F_A}{\partial \theta_A} &= kd\frac{\partial \hat{N}}{\partial \theta_A} + k\hat{N}\otimes\left(\frac{\partial d}{\partial \theta_A}\right)^T \\ &= -k\hat{N}\otimes(\widetilde{R_A r_A}R_A g) = k(\hat{N}\otimes R_A g)(\widetilde{R_A r_A})\end{aligned} \quad (33)$$

$$\begin{aligned}\frac{\partial F_A}{\partial \theta_B} &= kd\frac{\partial \hat{N}}{\partial \theta_B} + k\hat{N}\otimes\left(\frac{\partial d}{\partial \theta_B}\right)^T \\ &= -kd\tilde{\hat{N}} - k(\hat{N}\otimes R_A g)\widetilde{R_B r_B} \\ &= -\widetilde{F_A} - k(\hat{N}\otimes R_A g)\widetilde{R_B r_B}.\end{aligned} \quad (34)$$

By equations (19), we have

$$\frac{\partial F_B}{\partial y} = -\frac{\partial F_A}{\partial y}, \qquad \text{where } y \text{ stands for any variables.} \quad (35)$$

The vector product rule is

$$\frac{\partial}{\partial x}\Big(u(x)\times v(x)\Big) = \widetilde{u(x)}\frac{\partial v}{\partial x} - \widetilde{v(x)}\frac{\partial u}{\partial x}. \quad (36)$$

Based on equations (20) and (36), the gradients of contact torque can be derived as

$$\frac{\partial \tau_A}{\partial x_A} = \widetilde{R_A r_A}\frac{\partial F_A}{\partial x_A} + \widetilde{F_A}, \qquad \frac{\partial \tau_A}{\partial x_B} = \widetilde{R_A r_A}\frac{\partial F_A}{\partial x_B} - \widetilde{F_A}, \quad (37)$$

$$\frac{\partial \tau_A}{\partial \theta_A} = \widetilde{R_A r_A}\frac{\partial F_A}{\partial \theta_A}, \qquad \frac{\partial \tau_A}{\partial \theta_B} = \widetilde{R_A r_A}\frac{\partial F_A}{\partial \theta_B} + \widetilde{F_A}\widetilde{R_B r_B}, \quad (38)$$

$$\frac{\partial \tau_B}{\partial x_A} = \widetilde{R_B r_B}\frac{\partial F_B}{\partial x_A}, \qquad \frac{\partial \tau_B}{\partial x_B} = \widetilde{R_B r_B}\frac{\partial F_B}{\partial x_B}, \quad (39)$$

$$\frac{\partial \tau_B}{\partial \theta_A} = \widetilde{R_B r_B}\frac{\partial F_B}{\partial \theta_A}, \qquad \frac{\partial \tau_B}{\partial \theta_B} = \widetilde{R_B r_B}\frac{\partial F_B}{\partial \theta_B} + \widetilde{F_B}\widetilde{R_B r_B}. \quad (40)$$

4.2 Damping force and torque gradients

The damping forces F_{DA} and F_{DB}, which are related to relative velocity between contacting objects, are defined as follows:

$$F_{DB} = k_d(v_A + \omega_A\times R_A r_A - v_B - \omega_B\times R_B r_B), \quad (41)$$

$$F_{DA} = -F_{DB} = -k_d(v_A + \omega_A\times R_A r_A - v_B - \omega_B\times R_B r_B), \quad (42)$$

where k_d is the damping coefficient.

The derivatives of ω_A, ω_B to quaternion q_A, q_B can be computed separately for each of the components q_i of q.

$$\frac{\partial \omega_A}{\partial q_{iA}} = \Big(\frac{\partial R_A}{\partial q_{iA}}\bar{J}_A^{-1}R_A^T + R_A\bar{J}_A^{-1}\frac{\partial R_A}{\partial q_{iA}}^T\Big)L_A, \quad (43)$$

$$\frac{\partial \omega_A}{\partial q_{iB}} = 0, \qquad \frac{\partial w_B}{\partial q_{iA}} = 0, \quad (44)$$

$$\frac{\partial \omega_B}{\partial q_{iB}} = \Big(\frac{\partial R_B}{\partial q_{iB}}\bar{J}_B^{-1}R_B^T + R_B\bar{J}_B^{-1}\frac{\partial R_B}{\partial q_{iB}}^T\Big)L_B. \quad (45)$$

Quantities $\frac{\partial \omega}{\partial \theta}$ and $\frac{\partial \omega}{\partial q}$ are related as follows

$$\frac{\partial \omega}{\partial \theta} = \frac{\partial \omega}{\partial q}Q. \quad (46)$$

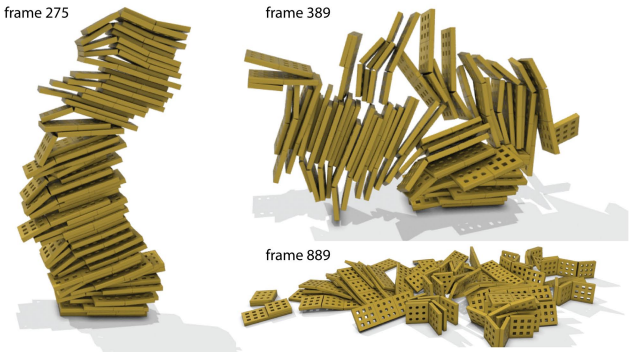


Fig. 6: Articulated rigid objects in complex contact and self-contact. *32 2-link and 32 3-link articulated objects (boxes with holes), 96 hinge joints. One simulation timestep takes 376 msec.*

Based on (41), the derivatives of F_{DB} are as follows:

$$\frac{\partial F_{DB}}{\partial x_A} = -k_d\widetilde{\omega_A}, \qquad \frac{\partial F_{DB}}{\partial x_B} = k_d\widetilde{\omega_A}, \quad (47)$$

$$\frac{\partial F_{DB}}{\partial \theta_A} = -k_d\widetilde{R_A r_A}\frac{\partial \omega_A}{\partial q_A}Q_A, \quad (48)$$

$$\frac{\partial F_{DB}}{\partial \theta_B} = k_d\Big((\widetilde{\omega_B} - \widetilde{\omega_A})\widetilde{R_B r_B} + \widetilde{R_B r_B}\frac{\partial \omega_B}{\partial q_B}Q_B\Big), \quad (49)$$

$$\frac{\partial F_{DB}}{\partial v_A} = k_d I, \qquad \frac{\partial F_{DB}}{\partial v_B} = -k_d I, \quad (50)$$

$$\frac{\partial F_{DB}}{\partial \omega_A} = -k_d\widetilde{R_A r_A}, \qquad \frac{\partial F_{DB}}{\partial \omega_B} = k_d\widetilde{R_B r_B}. \quad (51)$$

Using (35), we can now obtain the derivatives of F_{DA}. The derivatives of damping torques can be computed in a similar way to penalty contact torques.

5 Contact between Articulated Rigid Objects

We now extend our system to handle constraints between rigid objects (see Figure 6). We place no restriction on constraints which may form loops (Figure 2). We support three constraint types: ball and socket, hinge and prismatic joints. Each constraint k is specified by a constraint function $C_k \in \mathbb{R}^\ell$, which only depends on the center of mass position and rotation matrix of the two constrained objects, and ℓ depends on the constraint type. Specifically, our constraints are formulated as follows.

Ball and socket joint (pin constraint, $\ell = 3$) constrains two points, given by local coordinates X_i and X_j in each of the two constrained objects,

$$C_1(x_i, R_i, x_j, R_j) = x_i + R_i X_i - x_j - R_j X_j = 0. \quad (52)$$

Hinge joints ($\ell = 5$) are formed by using two ball and socket constraints for points $X_i^1, X_i^2, X_j^1, X_j^2$, where X_i^1 and X_i^2 are local coordinates, in frame of object i of two points on the hinge axis, and X_j^1 and X_j^2 are local coordinates

of those same points in frame of object j. This is mathematically equivalent to specifying constraints using X_i^1, X_j^1, and N, where N is the hinge rotation axis (expressed in the global coordinate system), and imposing that rotations R_i and R_j must transform N into the same vector. We use hinge joints in our boxes example, as well as with the excavator cabin, arms and bucket. Two rigid objects connected by a hinge joint have a total of 7 free degrees of freedom. One of the $2\times 3 = 6$ constraints is therefore redundant, and we remove it from C_2 as follows. We compute the current world coordinate direction of the hinge joint axis, $R_i(X_i^2 - X_i^1) \in \mathbb{R}^3$, and then find the dof (x, y, or z) of this vector with the largest absolute value. We can then arbitrarily choose either the first or second object (we choose second) to remove the scalar constraint corresponding to that dof.

Prismatic joint ($\ell = 8$) only permits translational motion of one rigid object relative to another. We use prismatic joints in the excavator example; as these joints translate, the excavator arms raise/lower or the bucket opens/closes. Two rigid objects constrained by a prismatic joint have 7 free degrees of freedom in total. So 3 of our 8 constraints are redundant but our SVD solver will help handle the rank-deficient system. We construct an orthogonal coordinate system, denoted as a_1, a_2, a_3, for the local frame of the prism, where a_1 is the unit vector in the direction joining center of mass positions of the two rigid objects i and j, $a_1 = (x_i - x_j)/||x_i - x_j||$. We set

$$C_3^1(x_i, R_i, x_j, R_j) = R_i(:,[1,2]) - R_j(:,[1,2]) = 0 \quad (53)$$

$$C_3^2(x_i, R_i, x_j, R_j) = (x_i - x_j)^T R_i[a_2, a_3] = [0,0], \quad (54)$$

where $R(:,[1,2])$ denotes the submatrix of the first 2 columns. Constraint $C_3^1 \in \mathbb{R}^6$ constrains two rigid objects to have the same rotation (note that matching the first two columns is sufficient), whereas $C_3^2 \in \mathbb{R}^2$ prevents any translation between the two objects except in the direction of a_1. We note that we first attempted to find an expression for the prismatic joint $C_3^*(x_1, x_2, R_1, R_2) = 0$ that only has five non-redundant constraints ($\ell = 5$). Such an approach would have to enforce $R_1 = R_2$ with three constraints. It would require conversion to a three-dimensional rotation representation such as Euler angles and would complicate constraint gradients.

We can assemble the constraints into a global constraint vector $C \in \mathbb{R}^m$,

$$C\Big(x_1, x_2, \ldots, x_n, R_1, R_2, \ldots, R_n\Big) = 0, \quad (55)$$

where m is the total number of constraints. To handle numerical drift, we use Baumgarte stabilization [27], where the acceleration constraint $\ddot{C} = 0$ is replaced by a linear combination of acceleration, velocity and position constraints

$$\ddot{C} + \alpha\dot{C} + \beta C = 0, \quad (56)$$

for some tunable scalar parameters $\alpha, \beta \geq 0$. We use critically damped stabilization, $\beta = \alpha^2/4$. We derive the first and second derivatives in Section 5.1,

$$\dot{C} = \sum_{i=1}^{n}\Big(\frac{\partial C}{\partial x_i} v_i + \frac{\partial C}{\partial R_i} : (\widetilde{\omega}_i R_i)\Big), \quad (57)$$

$$\ddot{C} = \sum_{i=1}^{n}\Big(\frac{\partial \dot{C}}{\partial x_i} v_i + \frac{\partial C}{\partial x_i} a_i + \frac{\partial \dot{C}}{\partial R_i} : (\widetilde{\omega}_i R_i) + \frac{\partial C}{\partial R_i} : (\dot{\widetilde{\omega}}_i R_i + \widetilde{\omega}_i^2 R_i)\Big), \quad (58)$$

where $\tilde{x}$ is the skew-symmetric matrix corresponding to vector $x \in \mathbb{R}^3$. Using (57) and (58), we can transform (56) into:

$$0 = \ddot{C} + \alpha\dot{C} + \beta C = \sum_{i=1}^{n}\Big(A_i\dot{P}_i + B_i\dot{L}_i\Big) + d, \quad (59)$$

where A_i, B_i, d are as follows (derivation in Section 5.1):

$$A_i = \frac{1}{m_i}\frac{\partial C}{\partial x_i},\ B_i = \hat{B}_i J_i^{-1},\ \hat{B}_i^{kl} = \frac{\partial C_k}{\partial R_i} : (\tilde{e}_l R_i), \quad (60)$$

$$d = \alpha\dot{C} + \beta C + \sum_{i=1}^{n}\Big(\frac{\partial \dot{C}}{\partial x_i} v_i + \frac{\partial \dot{C}}{\partial R_i} : \dot{\widetilde{\omega}}_i R_i + \frac{\partial C}{\partial R_i} : \widetilde{\omega}_i^2 R_i - B_i\dot{J}_i\omega_i\Big). \quad (61)$$

Here, C_k is the k-th element of $C \in \mathbb{R}^m$, and $e_\ell \in \mathbb{R}^3$ is the ℓ-th standard basis vector.

By combining (15) with the constraint forces $-A_i^T\lambda$ and torques $-B_i^T\lambda$, we obtain the following $\mathbb{R}^{(6n+m)\times(6n+m)}$ linear system:

$$\hat{G}\begin{bmatrix}\Delta S_1\\ \Delta S_2\\ \vdots\\ \Delta S_n\\ \lambda\end{bmatrix} = h\begin{bmatrix}Z_1\\ Z_2\\ \vdots\\ Z_n\\ d\end{bmatrix},\ \hat{G} = \begin{bmatrix}G & \mathcal{J}^T\\ \mathcal{J} & 0\end{bmatrix},\ \mathcal{J}^T = \begin{bmatrix}\mathcal{J}_1^T\\ \mathcal{J}_2^T\\ \vdots\\ \mathcal{J}_n^T\end{bmatrix}, \quad (62)$$

where λ is the constraint Lagrange multiplier, and

$$\mathcal{J}_i = [A_i \quad B_i]. \quad (63)$$

SVD solver: Because our constraints are formulated under maximal coordinates, some constraints in $\mathcal{J}$ are redundant, such as the rotation matching constraints in the prismatic joint (53). Therefore, the matrix $\hat{G}$ is singular, making it difficult to apply standard solvers directly to $\hat{G}$. The Kutzbach-Grübler equation [41] can be used to determine the *number* of redundant constraints. However, for a complex mechanism with loops, it is difficult to predict the redundant constraint *subspace*. Instead of attempting to enumerate the redundancies for their removal, we solve the Lagrange multiplier system (62) using a robust SVD solver that automatically removes the singularities. Such an approach works because the singularities can only enter (62) via $\mathcal{J}$. Therefore, we can safely form Schur's complement of G using the PARDISO sparse unsymmetric structurally-symmetric solver. When m is small, the inverse of $\hat{G}$ can be computed robustly and efficiently (see, e.g., [42]) as

$$\hat{G}^{-1} = \begin{bmatrix}G^{-1} + FDF^T & -FD\\ -DF^T & D\end{bmatrix}, \quad (64)$$

where $D = S^{-1} \in \mathbb{R}^{m\times m}$ for $S = -\mathcal{J}G^{-1}\mathcal{J}^T \in \mathbb{R}^{m\times m}$, and $F = G^{-1}\mathcal{J}^T \in \mathbb{R}^{6n\times m}$. Because $\mathcal{J}$ may be rank-deficient, we must compute D using a SVD-based pseudoinverse. We truncate all singular values smaller than $\varepsilon\sigma_0$, where $\varepsilon = 10^{-10}$ and σ_0 is the largest singular value of S. To solve (62), we do not explicitly form the inverse of $\hat{G}$, but solve the system by multiplying the right hand side with the matrix in (64), where multiplication by G^{-1} is performed by solving m linear systems (Section 4). Note that G corresponds to the penalty-based contact between objects, and is invertible. Pairs of constrained objects are put into the same contact island. Time complexity of solving Equation 62 is discussed in Section 5.2.

Active control is incorporated into our articulated system as follows. In our excavator example, for each prismatic joint, a PD servo motor adds forces F and $-F$ to the two constrained objects i and j, respectively, along the prismatic joint direction a_1. The PD force is

$$F = k_p\Big((x_j - x_i)^T R_i a_1 - L\Big)R_i a_1 + k_d\Big((v_j - v_i)^T R_i a_1\Big)R_i a_1, \tag{65}$$

where k_p is the stiffness gain, k_d is the damping gain, and L is the user-adjustable target distance between the center of mass positions x_i and x_j. Note that the L value directly affects the angles of the excavator joints. We limit L to an interval $[L_{\min}, L_{\max}]$, representing a reasonable workspace that our articulated system can reach. We integrate the PD controller using implicit integration. The gradient $\partial F/\partial S_i$ can be computed easily and assembled into terms $G_{i,i}, G_{i,j}, G_{j,i}, G_{j,j}, Z_i, Z_j$. An alternative, stiffer approach to controlling the excavator is to constrain the L with an additional constraint. We initially tried this approach, but the motion was overly stiff; whereas the PD controller (65) permits both stiff and soft gains, and can produce good-looking arm secondary motion. We employed a constraint to control the excavator cabin rotation, by directly specifying constraint $C_4 \in \mathbb{R}^6$:

$$C_4(x_i, R_i) = R_i(:,[1,2]) - R_i^*(:,[1,2]) = 0, \tag{66}$$

where R_i^* is the target rotation matrix for rigid object i.

5.1 Baumgarte stabilization derivation

In this section, we derive Equations 57, 58, 60, 61. The first and second derivative of the rotation matrix R are

$$\dot{R} = \tilde{\omega}R, \qquad \ddot{R} = \dot{\tilde{\omega}}R + \tilde{\omega}^2 R. \tag{67}$$

The constraint C is only related to the center of mass x and rotation matrix R, so by chain rule, we can arrive at (57) and (58). To incorporate this into our Lagrange multiplier system, we need to transform (56) into the specific format of (59), by reordering the terms. The undetermined coefficients A_i, B_i, d are given in (60), (61). Here is the verification:

$$A_i\dot{P}_i = \frac{\partial C}{\partial x_i}\frac{\dot{P}_i}{m_i} = \frac{\partial C}{\partial x_i}a_i, \tag{68}$$

$$\sum_{l=1}^{3} B_i^{kl}(\dot{\omega}_i)_l = \sum_{l=1}^{3}\Big(\frac{\partial C_k}{\partial R_i} : (\tilde{e}_l R_i)\Big)(\dot{\omega}_i)_l = \tag{69}$$

$$= \sum_{l=1}^{3}\frac{\partial C_k}{\partial R_i} : \Big((\dot{\omega}_i)_l \tilde{e}_l R_i\Big) = \tag{70}$$

$$= \frac{\partial C_k}{\partial R_i} : \Big((\sum_{l=1}^{3}(\dot{\omega}_i)_l \tilde{e}_l)R_i\Big) = \frac{\partial C_k}{\partial R_i} : (\tilde{\dot{\omega}}_i R_i). \tag{71}$$

$$L = J\omega, \qquad \dot{L} = \dot{J}\omega + J\dot{\omega}, \tag{72}$$

$$J = R\bar{I}R^T, \qquad \dot{J} = \tilde{\omega}J + J\tilde{\omega}^T = \tilde{\omega}J - J\tilde{\omega}, \tag{73}$$

$$\dot{\omega} = J^{-1}(\dot{L} - \dot{J}\omega) = J^{-1}(\dot{L} - (\tilde{\omega}J - J\tilde{\omega})\omega), \tag{74}$$

$$B_i\dot{L}_i = B_i J_i\dot{\omega}_i + B_i\dot{J}_i\omega_i = \hat{B}_i J_i^{-1}J_i\dot{\omega}_i + B_i\dot{J}_i\omega_i = $$

$$= \hat{B}_i\dot{\omega}_i + B_i\dot{J}_i\omega_i = \frac{\partial C}{\partial R_i} : (\tilde{\dot{\omega}}_i R_i) + B_i\dot{J}_i\omega_i. \tag{75}$$

The last term in (75) is canceled in d, and other remaining terms are also included in d, which is given as (61).

5.2 Complexity of incremental factorization

Complexity of solving Equation 62 is independent of the number of contacts. It depends on the severity of the contact configuration, the number of rigid objects n and the number of constraints m. In the extreme, unpractical case where every rigid body is in contact with every other body, G is dense and its factorization requires time $O(n^3)$. In the best case, the contact graph is a tree, and $Gx = y$ can be solved in time $O(n)$. In practice, the time to perform sparse Cholesky decomposition on G and backsubstitute to solve a linear equation, is $O(n^\alpha)$ (for $1 \le \alpha \le 3$) and $O(n^\beta)$ (for $1 \le \beta \le 2$), respectively. The complexity to solve Equation 62 using incremental factorization (Equation 64) is then

$$O(n^\alpha) + O(n^\beta)m + O(m^2 n) + O(m^3). \tag{76}$$

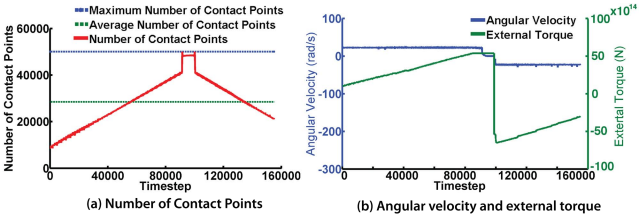


Fig. 7: Manipulating a hexbolt into the hole with controlled external torque and friction. *The hexbolt travels the entire length of the pre-existing thread on the hole. Bottom is reached at timestep 95,000. After 2 seconds, we reverse the torque and the hexbolt is unscrewed from the hole.*

6 Results

We present several examples demonstrating multiple rigid and articulated objects in complex distributed contact. All

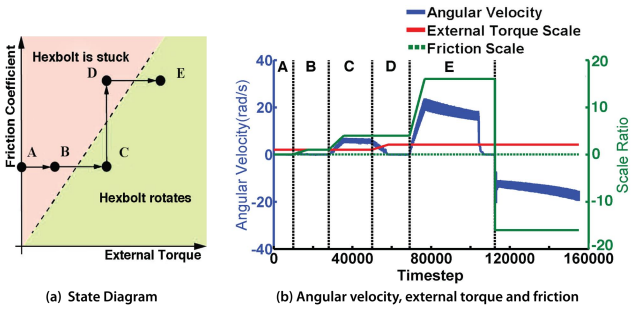


Fig. 8: Interplay of hexbolt friction and external torque: *Left: The state diagram.* **A***: external torque is zero, hexbolt is motionless;* **B***: external torque is too low, hexbolt is stuck;* **C***: external torque is increased, friction is overpowered and hexbolt rotates;* **D***: friction is increased, hexbolt is stuck again;* **E***: external torque increased, friction is overpowered again and hexbolt rotates. Right: angular velocity, external torque and friction.*

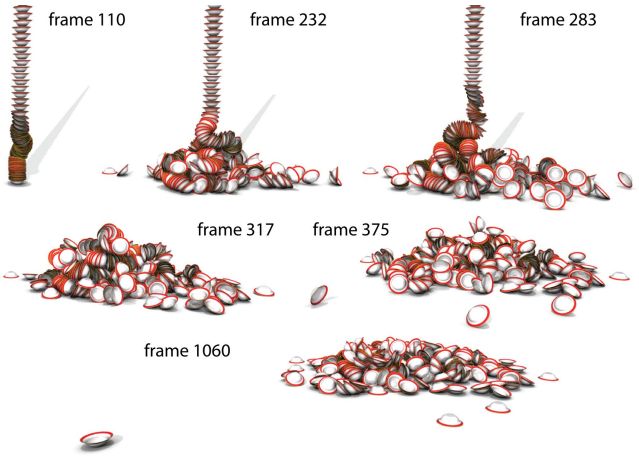


Fig. 9: 512 bowls in conforming distributed contact. *The bowls are dropped slightly off-center relative to each other. They form complex conforming harmonica-like patterns. They also roll on the ground and stack at the end.*

objects carry a pointshell and a signed distance field. The pointshell is either set to the vertices of the object, or precomputed using particle repulsion [35]. In all examples, the penetration depths and gradients are obtained by testing the first object's pointshell against the signed distance field of the second object [43], [35], then reversing the roles and testing the first object's signed distance field against the pointshell of the second object, and adding the results. We note that this approach produces a continuous force and torque, both in direction and magnitude, even when crossing the object's interior medial axis [35]. The reason for this is that we only get the force magnitude from the distance field, while the direction of the contact forces comes from the normal of the other contacting object (see [35]). Although the gradients are not continuous, the simulations remain stable in practice. Note that the contact gradients are also discontinuous everywhere at the surface as the force changes from zero force to a linearly growing force, yet this does not substantially impact simulation stability. We use an Intel Xeon 2.9 GHz CPU (2x8 cores) machine with 32GB RAM, and a GeForce GTX 680 graphics card with 2GB RAM.

In our first example (Figure 1), we use a screwdriver to stably screw a hexbolt into a threaded hole, with friction. We mimic the stabilizing effect of the human hand by removing 2 DOFs (rotation around x and z axis) from the screwdriver handle. To prevent sliding, we also add a PD control force that pushes position of screwdriver bit towards the center of the hexbolt's slit, mimicking the effect of human hand. This 4-body demo has a total of 16 DOFs: 6 for hexbolt and screwdriver bit each, 0 for the hole, and 4 for the screwdriver handle. Since the hexbolt's profile is serrated, the contact normals vary substantially in space. In the first hexbolt experiment, we manipulate the hexbolt the entire length of the hole, and back out, by driving the screwdriver with an external torque. The torque is PD-controlled so that the angular velocity is kept (approximately) constant (see Figure 7, right); note that the torque increases as the hexbolt sinks deeper into the whole. This example has a large number of contacts ($N = 45,000$, when the hexbolt is completely inserted, see Figure 7, left), and is feasible because our contact response time scales linearly with N : system preparation time is $O(N)$, whereas system solve time is independent of N. This example would be computationally difficult to simulate with other methods which typically scale superlinearly with N.

The second hexbolt experiment shows that the hexbolt motion is affected by the relative magnitudes of friction and external torque, thus generating complex but natural dynamics (Figure 8). We transition the object through five states (A-E), exploring the different combinations of friction and external torque.

Our second example demonstrates 512 bowls falling from the sky to the ground (Figure 9). The ground is static. The bowls generate large contact areas between each other, and the ground (maximum N is 8,500). Performance is analyzed in Figure 10. It can be seen that the system solve time is much smaller than collision detection time. One can observe the benefits of reducing the system into independent contact islands to the system solve time. System solve time increases as the bowls continue to stack onto each other, forming larger contact islands. We use a high-resolution distance field (1024^3) and a stiffness value that is able to prevent any pop-through bowl penetrations (see Figure 10, bottom-left).

To compare our method to the symplectic Euler method, we dropped 20 bowls from the same initial configuration with same simulation parameters (Figure 11). We found that the maximum stable timestep under our semi-implicit method is $10\times$ larger than the maximum stable timestep under the explicit method. Because the overhead of forming the implicit system and solving it, is small compared to collision detection, this translates into approximately $10\times$ faster computation time when timestep is increased, or

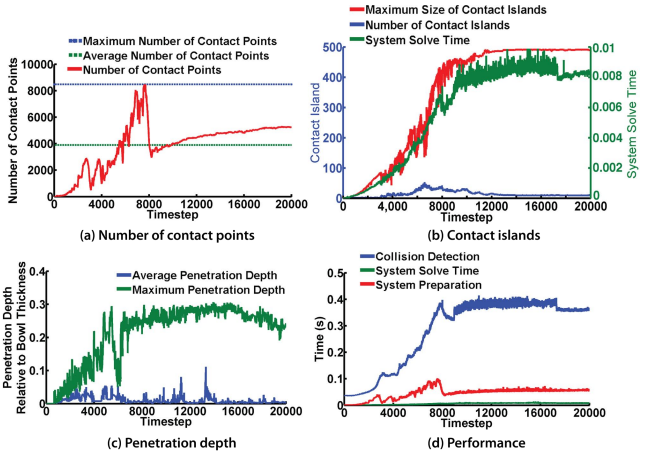


Fig. 10: 512 bowls simulation data analysis.

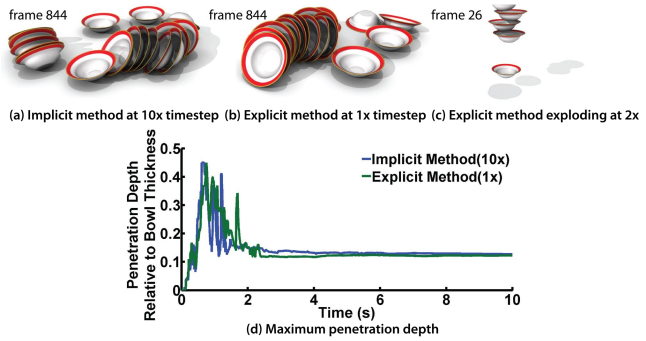


Fig. 11: Twenty bowls under explicit and semi-implicit methods. *Top: final configuration under the semi-implicit (10x timestep) and explicit (1x) method, as well as the exploding configuration at explicit (2x). 1x timestep is* $h = 6.7 \times 10^{-5}s$. *The explicit method is unstable for timesteps larger than 1x; we measured the ratio (using bisection) between max stable timestep to be 10x. Bottom: Maximum penetration depth. Same physical time for both curves.*

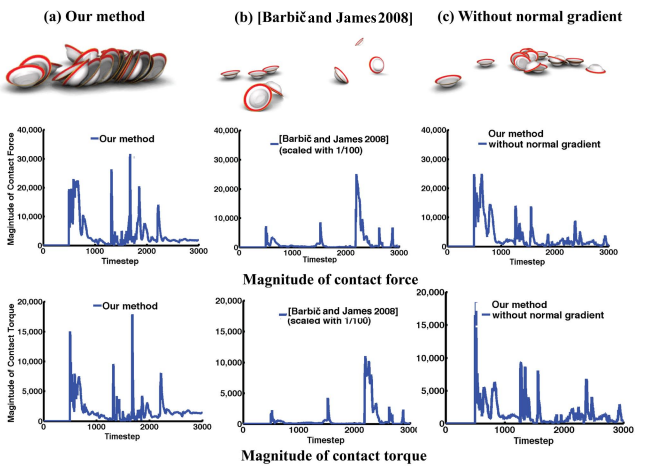


Fig. 12: Comparison of our analytical contact gradients to [35]. *The top row shows a representative frame (final frame in (a) and (c), and exploding in (b)). The middle and bottom row show the magnitude of contact forces and torques to the bowl that starts the simulation as the highest. It can be seen that our contact gradients (a) are more stable than those of [35] (b). In (c), we implemented a modification of our method where we removed gradients of contact forces and torques arising due to the rotating normals. The difference between (b) and (c) is that (c) employs the correct gradient of the distance field, as opposed to a normal-direction approximation [35]. It can be seen that the gradient quality greatly changes the course of the simulation. Simulation (b) is unstable (explodes), whereas (a) and (c) are significantly different.*

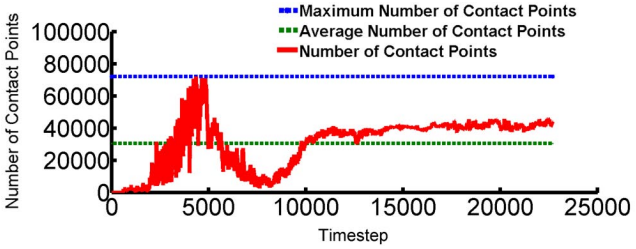


Fig. 13: Number of contact points (64 articulated objects).

stability gain at essentially no extra cost if timestep is kept the same. Figure 11 (bottom) demonstrates that there is little difference in penetration depth. In Figure 12, we compare our analytical contact gradients to [35].

Figure 6 shows 32 2-link and 32 3-link articulated objects with 96 hinge joints, consisting of 160 perforated boxes, falling to the ground (1,003K faces, 376 msec per timestep). Large maximum number of contacts (70,000) occurs (Figure 13). Maximum hinge joint position error across all timesteps was less than 10^{-7} the length of the box, implying that Baumgarte stabilization can prevent numerical drift effectively in our system.

The excavator example demonstrates a real-time (50 fps, including both simulation and rendering) controlled articulated system undergoing complex contact. The excavator (Figure 2) consists of 16 rigid parts, 4 prismatic joints and 17 hinge joints, forming 3 loops with non-trivial constraint redundancy. The user drives the excavator with the keyboard, lowering / raising the arms and opening / closing the bucket to pick up, transport and unload 9 rocks. We employ eight keyboard keys that set the target prismatic joint displacement for the three prismatic joint PD controllers, and the cabin hinge joint angle, which is modeled using a constraint. To limit the excavator to its natural workspace, we impose limits on target joint displacements. These limits also prevent excavator self-collisions, so there is no need to perform self-collision detection, although self-contact can be handled in our system (as in Figure 6). The 9 rocks are in contact with each other, the ground and the bucket, forming complex contact islands. We use our SVD solver to solve the rank-deficient system at every timestep. System solve time dominates our computation time, and increases when the maximum size of contact island increases (Figure 14, top-right, bottom-right). Our Baumgarte stabilization works well (Figure 14, top-left). The maximum constraint deviation for hinge joints occurs when the bucket hits the ground at a fast speed. Figure 14 (bottom-left) shows the prismatic joint movement

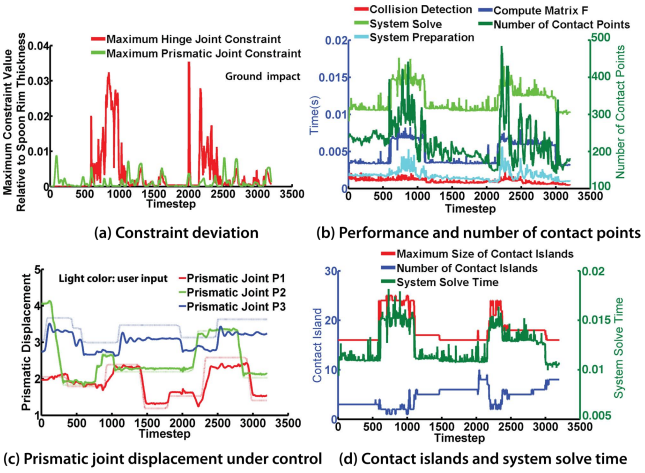


Fig. 14: Excavator simulation data analysis.

under the effects of gravity, contact and controlled motor force. It can be seen that our PD-controlled simulation provides interesting secondary motion in the prismatic joints, modeling the effects of mass inertia of the excavator links.

Stiffness scaling: In the excavator example, we found that we can avoid deep penetration depths if the contact stiffness is scaled as follows [33]:

$$k_{\text{modified}} = \begin{cases} k & \text{if } N < N_0, \\ kN_0/N & \text{if } N \geq N_0. \end{cases} \tag{77}$$

Here, k is the original contact stiffness. We denote $N_0 = \infty$ to mean that no change to k was performed. We used $N_0 = 10$ in the excavator example, and $N_0 = \infty$ in all other examples. To illustrate the effect of scaling, consider N parallel contacts with the ground plane. For $N < N_0$, the effective stiffness of all contacts together is kN, whereas for $N \geq N_0$, it is kN_0, i.e., clamped at a fixed maximum value independent of N. Such scaling is beneficial in scenarios where one must prevent deep penetration for small contacts ($N < N_0$), but limit the maximum stiffness under large contacts to maintain stability. Without the scaling, when grabbing a single rock, the rock would penetrate the excavator bucket, as the relatively few contacts on the rock together would not be strong enough to prevent penetration. If stiffness was increased to prevent rock penetration, however, the bucket would become unstable when it forms a flat large contact with the ground. For such scenarios, scaling makes it possible to have the best of both worlds: high stiffness for few contacts, and controlled maximum stiffness for large contact areas. The downside of the scaling is that the penalty force is no longer conservative, which can be mitigated using damping. We analyze the effect of parameter N_0 in Table 2. Although scaled simulations help with small contacts, it can be seen that they make it more difficult to keep the maximum penetration depth within a given threshold simulation-wide, when both small and large contacts are considered together: smaller timesteps are required to keep the same max penetration depths. This is because the scaling applies indiscriminately to all

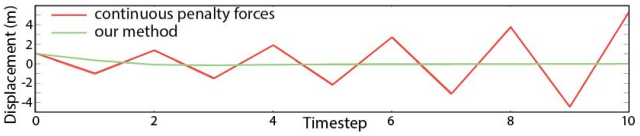


Fig. 15: Comparison of our method to continuous penalty forces. *Mass-spring system, unit mass, released with zero velocity from a unit displacement. We use a representative stiffness value where the timestep is too large for the explicit continuous penalty forces to remain stable.*

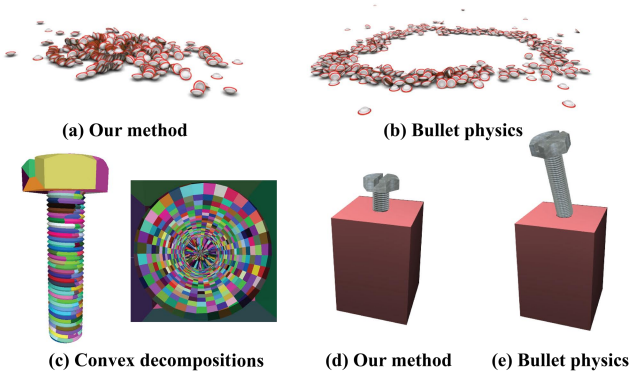


Fig. 16: Comparison to Bullet Physics. *(a,b) Final frame of the simulation using our method and Bullet Physics, respectively. (c) Convex decomposition generated by Bullet Physics. (d) Our method keeps the hexbolt stable in the hole. (e) Bullet Physics ejects the hexbolt.*

contacts. If the model has, for example, a large contact area and a smaller contact area, the global scaling will disproportionately weaken the smaller contact area and penetration will occur. To remedy such issues, it may be advantageous to use a spatially and temporally adaptive N_0. However, in examples with extreme contact, such as the hexbolt, high stiffness is required everywhere and at all times; scaling is not practical in such cases.

Comparison to continuous penalty forces: In Figure 15, we provide a comparison of our method to continuous penalty forces [20]. We do so by simulating a 1D mass-spring particle, by varying stiffness and observing system stability. Because the method of [20] is explicit, it becomes unstable if the spring stiffness exceeds the Courant condition as set by the simulation timestep. Our method, in turn, remains stable even for large stiffness values.

Parameter tuning can be a time-consuming task for penalty-based method, especially for large multi-body simulations with long computation times. In our examples, the parameters were determined hierarchically: we ran smaller simulations and tuned parameters, then re-used the same parameters for bigger simulations, which were then tweaked significantly less than the smaller simulations. We always tweaked using the initial part of simulations first, and then ran entire simulations. Table 1 shows the parameters and also the computation time for our examples.

Comparison to Bullet Physics: In Figure 16, we provide a comparison to the Bullet Physics [2] (v2.81) rigid body

	#timesteps per 1 sec of video	timestep[sec]	stiffness	video length	computation time
hexbolt (full-length insertion)	34,000	2.9E-5	2.96E13	47 sec	6.0 h
hexbolt	30,760	3.3E-5	2.42E13	52 sec	7.1 h
512 bowls	8,500	1.18E-4	26,732	27 sec	26.9 h
512 bowls (on-center)	9,560	1.05E-4	33,815	23 sec	29.0 h
boxes	10,660	9.4E-5	34,090	15 sec	16.7 h
excavator	50	2.0E-2	10,000	52 sec	52 sec

TABLE 1: Simulation parameters and statistics

N_0	largest stable timestep [sec]	max penetration depth (among bowls)	max penetration depth (with ground)	max #contact points
1	0.00001	0.192	0.950	2085
5	0.00003	0.192	0.575	2449
10	0.00001	0.194	0.972	2530
25	0.00002	0.192	0.787	1555
50	0.00003	0.177	0.894	886
100	0.0002	0.189	0.348	408
∞	0.0002	0.189	0.348	408

TABLE 2: Analysis of the stiffness scaling parameter N_0. *For each value of N_0, we tuned two parameters: contact stiffness k, and the timestep h. We tuned these two parameters manually, using bisection, so that the maximum penetration depth among the bowls closely matched the value at $N_0 = \infty$ (0.189) and timestep was as large as possible. Note that the larger the N_0, the more the simulation resembles the unscaled simulation ($N_0 = \infty$). It can be seen that as N_0 is increased, the simulation allows progressively larger timesteps (due to decreased stiffness), for equal or better penetration depths. 20-bowls example. Penetration depth is measured in units of the bowl thickness.*

simulation library. A comparison between LCP methods and penalty-based methods is challenging because there are many variants of LCPs, and because the LCP methods are exact, whereas penalty based methods are only approximate in avoiding penetration. LCP methods, however, are generally not designed for complex, conforming contact. We opted for Bullet Physics as a representative LCP method because it is widely used, works well in practice, and comes with available source code. Both Bullet and our method use single-core computation everywhere in this comparison. Bullet Physics provides two approaches to resolve collisions between non-convex rigid bodies: local penetration depth algorithm (btCompundShape from tet mesh), and convex decomposition followed by convex collision detection (btCompoundCollisionAlgorithm). We tried both approaches. The convex decomposition approach was significantly faster and we therefore use it in all of our experiments. In this approach, the input mesh is divided into convex pieces (see the hexbolt and hole in Figure 16, c). Bullet uses convex collision detection to identify contacts, followed by the resolution of contacts using a projected Gauss-Seidel LCP method. For both Bullet and our method, we found the largest stable timestep manually, using bisection. Our timestep was $3.5 \times 10^{-4} s$, whereas Bullet's timestep was $4.61 \times 10^{-3} s$ (13.1× bigger). Bullet's timesteps are, however, computationally 52× more expensive than ours in this example. With both Bullet and our method optimized to the best of our abilities, our total running time to produce animations of the same length (same total physical time) was therefore approximately 4× shorter than Bullet. Although both animations appear equally plausible, they differ visually (Figure 16, a, b). Our results are in line with the general observation that constraint-based approaches provide better accuracy (smaller penetration) for more computation. We also performed the same experiment for the hexbolt example. We inserted the hexbolt into the hole, and then started simulating the hexbolt under gravity. The expected correct result is that, after very brief small motion, the hexbolt stabilizes in the hole, near its initial position. We selected the largest timestep under which our method produces such a correct result (0.0333s). We normalized the methods by fixing the total physical time to be simulated. We then set the timestep for Bullet so that both methods dispensed an equal amount of total computational time (we excluded Bullet's convex decomposition time, which for the hexbolt took 1 hour). Because Bullet was computationally 2× slower in this example, this gave us a Bullet's timestep of $0.066s$. Our method produced a stable simulation, whereas Bullet Physics produced hexbolt vibrations, deep penetration into the sides of the hole and eventual ejection. We were able to keep the Bullet hexbolt stable by decreasing the Bullet timestep $20,000 \times (3.3 \times 10^{-6} s)$. This increased the Bullet computation time $70,000\times$ relative to our method. The increase is greater than $20,000\times$ because now, the hexbolt stays in the hole during the entire animation as opposed to being ejected soon after start, and therefore Bullet needs to resolve complex contact during the entire animation. We speculate that complex conforming contact is challenging for LCP methods because the discrete LCP search space is very high-dimensional, and the contacts change rapidly from timestep to timestep unless the timestep is small.

7 Conclusion

We presented an approach to simulate many rigid and articulated objects in complex conforming contact. We formulate constraints in maximal coordinates, and support loops. We show how articulated objects in distributed contact can

be actively controlled with PD servo controllers. We decrease computation time by performing symbolic Gaussian elimination, and robustly handle singular constraints using a SVD solver. Like most implicit methods, our method introduces artificial damping, and the damping grows higher with large timesteps and higher stiffness. Although such damping is sometimes desirable to slow down simulations (e.g., in computer games), methods that can both simulate stiff stable contact and keep damping under control, are left for future work. Although implicit, our method still requires tuning penalty stiffness parameters. Contact clustering [44] could be used to accelerate simulations; continuous distributed contact may benefit from geodesic-distance temporally coherent clustering to minimize penetrations and maximize the accuracy of dynamics. Our method supports local static and dynamic friction via "anchors". Although it can stably stack many objects, simulations where objects settle to rest in complex stacking configurations could be improved by a more precise friction model. Simulating active virtual characters in conforming contact with their environment, such as say, during training or maintenance of complex machinery, is important future work. We would also like to extend our method to distributed contact between multiple deformable objects. Finally, a more thorough survey and comparison between penalty-based and constraint methods for multibody dynamics with complex geometry would be beneficial for practitioners in this area.

ACKNOWLEDGMENTS

This research was sponsored in part by the National Science Foundation (CAREER-53-4509-6600), USC Annenberg Graduate Fellowship to Hongyi Xu, and a donation of two workstations by the Intel Corporation. We thank Erwin Coumans for help with the Bullet library.

REFERENCES

[1] J. Bender, K. Erleben, J. Trinkle, and E. Coumans, "Interactive simulation of rigid body dynamics in computer graphics," in *STAR Proceedings of Eurographics*, 2012.

[2] E. Coumans, "Bullet Physics," 2012, http://www.bulletphysics.com.

[3] D. Baraff, "Fast contact force computation for nonpenetrating rigid bodies," in *Proc. of ACM SIGGRAPH 94*, 1994, pp. 23–34.

[4] D. Stewart and J. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction," *Int. J. of Numerical Methods in Engineering*, vol. 39, pp. 2673–2691, 1996.

[5] D. Baraff, "An introduction to physically based modeling: Rigid body simulation i: unconstrained rigid body dynamics," in *SIGGRAPH '97 Course Notes*, 1997, p. 97.

[6] C. Duriez, F. Dubois, A. Kheddar, and C. Andriot, "Realistic Haptic Rendering of Interacting Deformable Objects in Virtual Environments," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 12, no. 1, pp. 36–47, 2006.

[7] R. Tonge, F. Benevolenski, and A. Voroshilov, "Mass splitting for jitter-free parallel rigid body simulation," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 105:1–105:8, 2012.

[8] B. Mirtich, "Impulse-based Dynamic Simulation of Rigid Body Systems," Ph.D. dissertation, Univ. of California at Berkeley, 1996.

[9] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically Deformable Models," *Computer Graphics (Proc. of ACM SIGGRAPH 87)*, vol. 21(4), pp. 205–214, 1987.

[10] D. M. Kaufman, T. Edmunds, and D. K. Pai, "Fast frictional dynamics for rigid bodies," in *ACM SIGGRAPH 2005 Papers*, ser. SIGGRAPH '05. New York, NY, USA: ACM, 2005, pp. 946–956.

[11] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino, "Collision Detection for Deformable Objects," *Computer Graphics Forum*, vol. 24, no. 1, pp. 61–81, 2005.

[12] S. Fisher and M. C. Lin, "Fast penetration depth estimation for elastic bodies using deformed distance fields," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2001, pp. 330–336.

[13] B. Heidelberger, M. Teschner, R. Keiser, M. Mller, and M. H. Gross, "Consistent penetration depth estimation for deformable collision response," in *VMV'04*, 2004, pp. 339–346.

[14] 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*, may 2006, pp. 1904 –1909.

[15] R. E. Ellis, N. Sarkar, and M. A. Jenkins, "Numerical methods for the force reflection of contact," *Journal of Dynamic Systems, Measurement, and Control*, vol. 119, no. 4, pp. 768–774, 1997.

[16] S. Hasegawa and M. Sato, "Real-time rigid body simulation for haptic interactions based on contact volume of polygonal objects," *Computer Graphics Forum*, vol. 23, pp. 529–538, 2004.

[17] E. Drumwright, "A fast and stable penalty method for rigid body simulation," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 14, no. 1, pp. 231 –240, 2008.

[18] D. Harmon, E. Vouga, B. Smith, R. Tamstorf, and E. Grinspun, "Asynchronous Contact Mechanics," *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 87:1–87:12, 2009.

[19] D. Harmon, Q. Zhou, and D. Zorin, "Asynchronous integration with phantom meshes," in *Symp. on Computer Animation (SCA)*, 2011, pp. 247–256.

[20] M. Tang, D. Manocha, M. A. Otaduy, and R. Tong, "Continuous penalty forces," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 107:1–107:9, 2012.

[21] D. Baraff and A. Witkin, "Physically based modelling, ACM SIGGRAPH Course Notes," 2001.

[22] E. Celledoni and B. Owren, "Lie group methods for rigid body dynamics and time integration on manifolds," *Computer Methods in Applied Mechanics and Engineering*, vol. 192, no. 34, pp. 421–438, 2003.

[23] M. Kobilarov, K. Crane, and M. Desbrun, "Lie Group Integrators for Animation and Control of Vehicles," *ACM Trans. on Graphics*, vol. 28, no. 2, pp. 1–14, 2009.

[24] R. Featherstone, *Robot Dynamics Algorithms*. Boston: Kluwer Academic Publishers, 1987.

[25] A. Witkin, M. Gleicher, and W. Welch, "Interactive dynamics," in *Computer Graphics (Proc. of SIGGRAPH 90)*. ACM SIGGRAPH, 1990, pp. 11–21.

[26] R. Weinstein, J. Teran, and R. Fedkiw, "Dynamic simulation of articulated rigid bodies with contact and collision," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 3, pp. 365–374, May 2006.

[27] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 1, no. 1, pp. 1–16, 1972.

[28] D. Baraff and A. P. Witkin, "Large Steps in Cloth Simulation," in *Proc. of ACM SIGGRAPH 98*, July 1998, pp. 43–54.

[29] M. A. Otaduy, "6-DoF Haptic Rendering Using Contact Levels of Detail and Haptic Textures," Ph.D. dissertation, Department of Comp. Science, University of North Carolina at Chapel Hill, 2004.

[30] M. A. Otaduy and M. C. Lin, "Stable and Responsive Six-Degree-of-Freedom Haptic Manipulation Using Implicit Integration," in *Proc. of the World Haptics Conference*, 2005, pp. 247–256.

[31] M. A. Otaduy, R. Tamstorf, D. Steinemann, and M. H. Gross, "Implicit contact handling for deformable objects," *Comput. Graph. Forum*, pp. 559–568, 2009.

[32] F. Hernández, C. Garre, R. Casillas, and M. Otaduy, "Linear-time dynamics of characters with stiff joints," in *V Ibero-American Symp. in Computer Graphics (SIACG 2011)*, 2011.

[33] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, "Six degree-of-freedom haptic rendering using voxel sampling," in *Proc. of ACM SIGGRAPH 99*. ACM, 1999, pp. 401–408.

[34] W. McNeely, K. Puterbaugh, and J. Troy, "Voxel-Based 6-DOF Haptic Rendering Improvements," *Haptics-e*, vol. 3, no. 7, 2006.

[35] J. Barbič and D. L. James, "Six-dof haptic rendering of contact between geometrically complex reduced deformable models," *IEEE Transactions on Haptics*, vol. 1, no. 1, pp. 39–52, 2008.

[36] F. Faure, S. Barbier, J. Allard, and F. Falipou, "Image-based collision detection and response between arbitrary volume objects," in *Symp. on Computer Animation (SCA)*, 2008, pp. 155–162.

[37] J. Allard, F. Faure, H. Courtecuisse, F. Falipou, C. Duriez, and P. G. Kry, "Volume contact constraints at arbitrary resolution," *ACM Trans. on Graphics (SIGGRAPH 2010)*, vol. 29, no. 3, pp. 82:1–82:10, 2010.

[38] B. Wang, F. Faure, and D. K. Pai, "Adaptive image-based intersection volume," *ACM Trans. on Graphics (SIGGRAPH 2012)*, vol. 31, no. 4, pp. 97:1–97:9, 2012.

[39] E. Larsen, "A Robot Soccer Simulator: A Case Study for Rigid Body Contact, Sony Computer Entertainment America R&D," Sony Computer Entertainment America R&D, Tech. Rep., 2001.

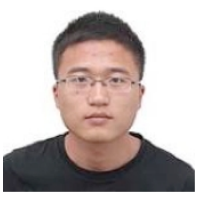
[40] O. Schenk and K. Gärtner, "Solving unsymmetric sparse systems of linear equations with PARDISO," *Future Generation Computer Systems*, vol. 20, no. 3, pp. 475–487, 2004.

[41] M. Grübler, "Allgemeine Eigenschaften der Zwangläufigen ebenen kinematischen Ketten, Part I," *Zivilingenieur*, vol. 29, pp. 167–200, 1883.

[42] J. Barbič, F. Sin, and E. Grinspun, "Interactive editing of deformable simulations," *ACM Trans. on Graphics (SIGGRAPH 2012)*, vol. 31, no. 4, 2012.

[43] J. Barbič, "Real-time reduced large-deformation models and distributed contact for computer graphics and haptics," Ph.D. dissertation, Carnegie Mellon University, Aug. 2007.

[44] Y. J. Kim, M. A. Otaduy, M. C. Lin, and D. Manocha, "Six degree-of-freedom haptic display using incremental and localized computations," *Presence-Teleoperators and Virtual Environments*, vol. 12, no. 3, pp. 277–295, 2003.



Hongyi Xu is a PhD student in computer science at the University of Southern California. He obtained his BS degree from Zhejiang University. His research interests are in computer graphics, physically based animation, contact and interactive physics.

Yili Zhao is a PhD student in computer science at the University of Southern California. He obtained his BS and MS degrees from Nanjing University of Aeronautics and Astronautics and Peking University, respectively. His research interests are in computer graphics, physically based animation and botanical simulation.

Jernej Barbič is an assistant professor of computer science at USC. In 2011, MIT Technology Review named him one of the Top 35 Innovators under the age of 35 in the world (TR35). Jernej's research interests include nonlinear solid deformation modeling, model reduction, collision detection and contact, optimal control and interactive design of deformations and animations. He is the author of Vega FEM, an efficient free C/C++ software physics library for deformable object simulation. Jernej is a Sloan Fellow (2014).