Technical Supplement to Incremental Potential Contact: Intersection- and Inversion-free, Large-Deformation Dynamics

MINCHEN LI, University of Pennsylvania & Adobe Research
ZACHARY FERGUSON and TESEO SCHNEIDER, New York University
TIMOTHY LANGLOIS, Adobe Research
DENIS ZORIN and DANIELE PANOZZO, New York University
CHENFANFU JIANG, University of Pennsylvania
DANNY M. KAUFMAN, Adobe Research

ACM Reference Format:
Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. 2020. Technical Supplement to Incremental Potential Contact: Intersection- and Inversion-free, Large-Deformation Dynamics. *ACM Trans. Graph.* 39, 4, Article 49 (July 2020), 6 pages. https://doi.org/10.1145/3386569.3392425

Contents

1 SMOOTHING

Let $f(x)$ be a function we wish to smooth. It is C^1 continuous everywhere except at $x = a$ where it is only C^0 continuous. Applying a function $g(x)$ that is C^1 continuous everywhere with $g(a) = 0$, we have a smoothed function $f(x)g(x)$ that is C^1 continuous everywhere. For $x \neq a$ we have $(f(x)g(x))' = f'(x)g(x) + f(x)g'(x)$ is C^0 continuous everywhere. At $x = a$, the left and right derivatives of $f(x)g(x)$ are then

$$\begin{aligned}\lim_{x\to a^-}(f(x)g(x))' &= \lim_{x\to a^-} f'(x)g(a) + f(a)g'(a),\\ \lim_{x\to a^+}(f(x)g(x))' &= \lim_{x\to a^+} f'(x)g(a) + f(a)g'(a).\end{aligned} \tag{1}$$

As $f(x)$ is C^0 continuous at $x = a$, $\lim_{x\to a^-} f'(x)$ and $\lim_{x\to a^+} f'(x)$ are both bounded. Then with $g(a) = 0$ we then have left and right derivatives of $f(x)g(x)$ both equal $f(a)g'(a)$ at $x = a$,

$$\lim_{x\to a^-}(f(x)g(x))' = \lim_{x\to a^+}(f(x)g(x))' = f(a)g'(a) \tag{2}$$

Thus $(f(x)g(x))'$ is likewise C^0 continuous at $x = a$ and $f(x)g(x)$ is correspondingly C^1 continuous.

2 BARRIER CONTINUITY AND TESTING

The continuity of our C^2 barrier function is confirmed when $d < \hat{d}$, as $\frac{\partial b}{\partial d} = (\hat{d} - d)(2\ln\frac{d}{\hat{d}} - \frac{\hat{d}}{d} + 1)$ and $\frac{\partial^2 b}{\partial d^2} = -2\ln\frac{d}{\hat{d}} + (\hat{d} - d)\frac{\hat{d}+3d}{d^2}$ both vanish as $d \to \hat{d}$. Thus the left and right derivatives of b at $d = \hat{d}$ are both equal at zero up to 2nd order.

Our motivation for applying a C^2 clamped barrier rather than a less nonlinear, but still smooth, C^1 barrier is to provide 2nd-order derivatives suitable for our Newton-type solver. Thus it is genralluy better to have a continuous Hessian for improved convergence [Nocedal and Wright 2006]. Nevertheless, here we also provide an ablation study applying all both C^0 ($b = -\ln(d/\hat{d})$) and C^1 ($b = (d - \hat{d})\ln(d/\hat{d})$), along with our final choice of our C^2 ($b = -(d - \hat{d})^2\ln(d/\hat{d})$) barrier in IPC on a set of examples in Figure 1.

From the results we see that for the C^0 barrier, optimization can be non-convergent. Here this can be a result if the local minima is right inside the clamped region of the barrier, where the gradient does not change smoothly – here intermediate values may not be found to balance terms so decrease the total gradient. While, in comparison to C^1, our C^2 barrier is generally 5%–10% faster due to the continuity of the Hessian – this is reflected in less iteration counts.

3 CFL-INSPIRED CULLING OF CCD

As IPC requires performing CCD for every Newton iteration, CCD clearly becomes a bottleneck. Therefore we propose a novel CFL-inspired culling strategy to accelerate CCD.

Authors' addresses: Minchen Li, University of Pennsylvania & Adobe Research, minchernl@gmail.com; Zachary Ferguson; Teseo Schneider, New York University; Timothy Langlois, Adobe Research; Denis Zorin; Daniele Panozzo, New York University; Chenfanfu Jiang, University of Pennsylvania; Danny M. Kaufman, Adobe Research, dannykaufman@gmail.com.

0730-0301/2020/7-ART49 $15.00
https://doi.org/10.1145/3386569.3392425

Examples	C0 # iters, t (s) / time	C1 # iters, t (s) / time	C2 (IPC) # iters, t (s) / time
mat on knives	5.51, 2.16	5.59, 1.43	5.47, 1.40
2 mats40x40 fall	NC	26.83, 10.64	26.17, 10.48
octocat on knives	NC	9.29, 4.21	7.73, 3.76
sphere1K roller	NC	47.64, 9.41	45.22, 9.02
mat40x40 twist (10s)	7.74, 1.93	8.04, 2.07	7.56, 1.89
sphere1K pin-cushion	12.82, 1.27	9.22, 0.75	8.93, 0.69
rods630 twist (10s)	6.65, 1.05	3.05, 0.38	3.07, 0.40

Fig. 1. **Ablation study on barrier functions with different continuity.** NC means not converging after 10000 PN iterations when solving a certain time step.

Recall that our culled constraint set contains all surface primitive pairs with distances smaller than $\hat{d}$. Thus all remaining primitives outside the culled constraint set have farther distances lower-bounded by $\hat{d}$. We place all vertices participating in these primitives in a set $\mathcal{F}$. At each iteration i with a search direction p^i, we find the index ℓ of the simulation node with the largest component in p (i.e., its effective search-direction "velocity") among all remaining node pairs, $\ell = \text{argmax}_{k\in\mathcal{F}}||p_k^i||$. We then compute a conservative bound on the largest-feasible step size for surface primitives not in $\hat{\mathcal{C}}$ as

$$\alpha_{\mathcal{F}} = \frac{\hat{d}}{2||p_\ell^i||}. \tag{3}$$

We then apply conservative CCD (see below) to the remaining primitive pairs in $\hat{\mathcal{C}}(x^i)$, obtaining a large feasible step size, $\alpha_{\hat{\mathcal{C}}}$, for that set. Then $\alpha_0 = \min(\alpha_{\mathcal{F}}, \alpha_{\hat{\mathcal{C}}})$ is our maximum *feasible* step size for iteration i. We then apply α_0 as starting step size to bootstrap backtracking for Newton iteration i and so ensure decrease with feasibility.

Computing large, feasible step sizes in this way effectively reduces *CCD* cost by two-orders of magnitude. However, for scenes that contain high speed motions, $\alpha_{\mathcal{F}}$ can be overly conservative (smaller than needed) which then would increase iteration counts and so cause energy related computations to increase overall cost unnecessarily.

Thus we adapt by balancing between applying full CCD for all candidate pairs provided by spatial hash and applying our CFL-like strategy. Here we will designate the exact feasible bound computed from applying CCD to all primitive pairs in the spatial hash as $\alpha_{\mathcal{S}}$.

In each step we first compute $\alpha_{\mathcal{F}}$ and $\alpha_{\hat{\mathcal{C}}}$ – they are both efficient and inexpensive to find. Next we observe that the culled bound $\alpha_{\hat{\mathcal{C}}}$ is often very close to the exact bound $\alpha_{\mathcal{S}}$, while computing this exact bound is generally one-third of the total timing cost in a single iteration. We thus proceed by computing $\alpha_{\mathcal{S}}$ if $\alpha_{\mathcal{F}} < \frac{1}{2}\alpha_{\hat{\mathcal{C}}}$. Otherwise we apply our CFL-type bound and apply $\alpha_0 = \min(\alpha_{\mathcal{F}}, \alpha_{\hat{\mathcal{C}}})$.

We thus avoid overly restrictive step sizes when $\alpha_{\mathcal{F}}$ and $\alpha_{\hat{\mathcal{C}}}$ are already quite close – meaning their minimum should also be quite close to $\alpha_{\mathcal{S}}$. In practice we observe that our CFL-like assisted CCD culling strategy provides a 50% speed-up for all CCD related costs with nearly the same iteration counts. Ovxerall this results in an average 10% speedup for IPC; see Figure 2.

Examples	full CCD # iters, t (s) / time step	CFL combined (IPC) # iters, t (s) / time step	full CCD CCD related Timing (s)	CFL combined (IPC) CCD related Timing (s)
mat on knives	5.34, 1.66	5.47, 1.40	97.50	47.25
2 mats40x40 fall	23.22, 10.21	26.17, 10.48	273.38	162.10
octocat on knives	6.99, 3.92	7.73, 3.76	206.61	129.93
sphere1K roller	49.38, 9.89	45.22, 9.02	482.13	394.01
mat40x40 twist (10s)	7.37, 2.13	7.56, 1.89	147.60	62.91
sphere1K pin-cushion	8.35, 0.77	8.93, 0.69	37.25	19.61
rods630 twist (10s)	3.04, 0.47	3.07, 0.40	23.89	9.13

Fig. 2. **CCD strategies ablation.**

4 CONSERVATIVE CCD

CCD is generally applied to compute a time of impact corresponding to a step size that would bring distances between primitives to 0. In our barrier setting this "largest feasible step size" needs to be made conservative by backing away from an exact zero distance. A simple strategy would be a conservative rescaling with a factor $c \in (0, 1)$; e.g., by starting the line search at 0.5 or 0.9 of the total step along the descent direction. However, for the CCD computations rounding error can be severe for the tiny contacting distances we allow and so even small naive scaling factors (e.g., 0.1) can allow unacceptable intersections in such cases while in others is a much too conservative bound unnecessarily slowing convergence.

Rather than directly finding and then conservatively rescaling a CCD-computed step length that takes us to intersection we directly compute via CCD a step size along p^i that will bring primitives to a distance of $(1-c)d_c > 0$. Here d_c is the current distance between the primitives. Standard CCD libraries [1] directly provide this option, e.g., exposed as an η parameter. In turn this modifies coefficients of the polynomial equations solved during CCD, and effectively reduces numerical rounding errors that cause issues for direct scaling. In our implementation we apply $c = 0.8$ between mesh primitives, $c = 0.9$ for mesh-to-plane, and similarly $c = 0.8$ for computing the large feasible step size to avoid element inversion when barrier elasticity energies, e.g., neo-Hookean, are applied.

Finally, to ensure we remain intersection- and inversion- free at each step. We apply a post-step check whenever nodal positions are displaced (e.g. line search and initial movements of boundary conditions and collision objects. These include the edge-triangle intersection check filtered by our spatial hash and a volume check for every tetrahedral element. In exceedingly rare cases when an edge-triangle intersection or negative volume are detected, we half the final step size bound.

[1] we use https://github.com/evouga/collisiondetection

5 EQUALITY CONSTRAINTS FOR MOVING COLLISION OBJECTS AND TIME-VARYING BOUNDARY CONDITIONS

In many scenarios, e.g., scripting animations, kinematic objects, and engineering tasks, scripted kinematic collision objects (CO) and/or moving positional (Dirichlet-like) boundary conditions (BC) are required. Contact algorithms generally handle these functionalities by either directly prescribing and updating nodal positions of CO/BC nodes at start of each time step, or interpolating them in substeps across a step. Remainder of simulation DOF are then solved w.r.t. the prescribed nodes being fixed at "current" positions. However, such strategies are extremely limiting, with simulations generally restricted to small time step sizes, speeds, and/or deformations as the BO and/or CO become faster and more challenging. For example, directly prescribing CO or BC nodes can often generate tunneling artifacts, and for moving BC, simulations can often fail simply by inverting elements when barrier energies like neo-Hookean energy are applied. To address these issues we formulate scripted dynamic BC and CO as equality constraints in IPC. This simultaneously ensures that intersections (and inversions) are avoided even while scripted motions are applied at large time step.

We start by computing the prescribed nodal positions at the beginning of each time step, and then apply CCD and element inversion detection to find a large feasible step size towards the prescribed position from the current one (see above).

If it is safe to apply them fully without causing intersection or inversion, we simply update prescribed nodal positions, solve the remainder of simulation DOF and are done. However, if our feasible step size is smaller than taking a full step (we use a criterion of 0.999), we first move the prescribed nodes as far as we can conservatively (see Section 4) and then add new *equality constraints* for each of these prescribed nodes provided by either CO or BC scripting.

As in our treatment of intersection-free *inequality* constraints, we build an unconstrained form for each by applying an augmented Lagrangian. For every such prescribed node, we add a Lagrangian and a penalty potential. We initialize each Lagrange multiplier to 0 and each penalty stiffness to 10^6. Concretely, we add an energy

$$-\sqrt{m_k}\lambda_{\mathrm{A},k}^T(x_k - \hat{x}_k) + \frac{\kappa_\mathrm{A}}{2}m_k||x_k - \hat{x}_k||^2 \tag{4}$$

to our barrier-form incremental potential for each prescribed node k with corresponding destination $\hat{x}_k$ in the current time step, if *any* of the prescribed nodes could not reach its destination during our start-of-time-step test.

We measure satisfaction of BC/CO node constraints at each Newton iteration i by calculating their total in-time-step progress as

$$\eta_\mathrm{A} = 1 - \sqrt{\frac{\sum_k ||\hat{x}_k^{t+1} - x_k^i||^2}{\sum_k ||\hat{x}_k^{t+1} - x_k^t||^2}}, \tag{5}$$

where $\hat{x}^{t+1}$ is the prescribed BC/CO positions for time step $t+1$. Then, whenever a current iterate is both close to satisfying stationarity, via the stopping criteria, and progress, with $\eta_\mathrm{A} < 0.999$, we either increase the BC/CO penalty stiffness or else update the Lagrange multipliers, via the first-order update rule, see Algorithm 1 below. Finally, whenever $\eta_\mathrm{A} \geq 0.999$, we fix all prescribed nodes at current position and solve for remaining DOF in order to avoid unnecessary slow down of convergence due to added stiffness.

Algorithm 1 Augmented Lagrangian Update Rule

for each PN iteration i **do**
  ...
  if $\frac{1}{h}||p^i||_\infty < \max(10^{-2}l, \epsilon_d)$ **and** $\eta_\mathrm{AL} < 0.999$ **then**
    if $\eta_\mathrm{AL} < 0.99$ **and** $\kappa_\mathrm{AL} < 10^8$ **then**
      $\kappa_\mathrm{AL} \leftarrow 2\kappa_\mathrm{AL}$
    else
      for each constrained node k **do**
        $\lambda_{\mathrm{AL},k} \leftarrow \lambda_{\mathrm{AL},k} - \kappa_\mathrm{AL}\sqrt{m_k}(x_k^i - \hat{x}_k^{t+1})$

For codimensional surface and segment collision objects their nodal mass is computed by estimating their nodal volume as the half sphere with diameter being the average length of the incident edges. For point collision objects we set their mass to be the average nodal mass of the simulated objects. Alternatively, simply setting all codimensional nodal masses to be the average of the simulated objects should also be fine. These estimated masses are only used for moving codimensional collision objects and they do not affect any physical accuracy.

6 ADAPTIVE BARRIER STIFFNESS

Stiffness, and so difficulty in solution of the barrier comes from two sources: $\hat{d}$ and likewise κ. As we can improve accuracy by directly decreasing $\hat{d}$, this frees κ to adaptively condition our solves for improved convergence.

A feature of our barrier framework is that the estimation of Lagrange multipliers are efficiently self-adjusted by the constraint values, in our case, the d_k's. However, if κ is not set appropriately, the contact primitives would either need to get extremely close to get enough repulsion (barrier gradient) when κ is too small, or need to get a distance right below $\hat{d}$ for a small repulsion if κ is set too large, both resulting in slow convergence because of ill-conditioning and nonsmoothness. Thus adaptively setting and/or adjusting κ is essential.

Intuitively, the smaller d_k is, the better the complimentarity condition is satisfied. This is certainly true from optimization theory, but is troublesome in numerical computation. In numerical optimization, since double precision floating point number is used, when d_k gets tiny, the rounding errors will significantly slow down convergence and even result in incorrect intermediate computations. Therefore, we must also prevent d_k from being too tiny.

If we view $\hat{d}$ as a control on the upper bound acting distance of our contact forces, κ can be seen as an indirect control on the lower bound of the acting distance as larger κ can provide the same amount of "repulsion" at a larger distance, avoiding the need to push distances to become too tiny. Tiny distances not only make CCD less robust and make the optimization less efficient in our numerical simulation, but also are not physically reasonable in science. Generally speaking, the space between the nucleus of two bonded atoms is around 10^{-10} meters. There is no way for two macroscopic touching objects to get to that close. On the other hand,

recall that the energy gradient of our optimization is

$$g(x,\kappa,\hat{d}) = \nabla E(x) + \kappa \sum_k \frac{\partial b}{\partial d_k}\nabla d_k(x,\hat{d}), \tag{6}$$

where at subproblem convergence, the IP gradient $\nabla E(x)$ balances with the barrier gradient $\kappa \sum_k \frac{\partial b}{\partial d_k}\nabla d_k(x,\hat{d})$. In addition, the barrier stiffness κ also influences the condition of the energy Hessian. Thus we adapt barrier stiffness strategy based on balancing the two gradients, iteratively increasing κ when needed, and applying lower and upper bounds obtained from conditioning analysis on the Hessian. Here d_k can then avoid being too small or too close to $\hat{d}$ to provide improved convergence regardless of $\hat{d}$, material, h, and our other input settings change.

The idea of balancing gradients can be traced back in optimization literature [Nocedal and Wright 2006] for estimating appropriate initial stiffnesses of barriers. In our case, we solve $\text{argmin}_\kappa ||g(x^j,\kappa,\hat{d})||^2$ which gives us $-g_c \cdot \nabla E(x)/||g_c||^2$ (where $g_c = \sum_k \frac{\partial b}{\partial d_k}\nabla d_k(x,\hat{d})$) at start of each time step to obtain an estimate of κ that seeks to balance the two gradients at x^j. However, we observe that the effectiveness of this balancing term is highly dependent on the x^j applied. It can be quite far from the configuration at solution and so potentially can leads to poorly scaled or even negative values for κ. Thus, we extend our analysis from the balancing gradients to additionally include conditioning of the Hessian. This, in turn, obtains an effective estimate to provide a lower bound of κ in support of the gradient balancing strategy.

Our analysis seeks to keep the scaling of the diagonal entries of the Hessian, at small distances, close to the mass. Specifically, a scaling characterization of the Hessian diagonal in $\nabla^2 b$ at $d = 10^{-8}l$ is easily estimated by taking its first term $\nabla d^T \frac{\partial^2 b}{\partial d^2}\nabla d$ in point-point formula as

$$c_{\nabla^2 b} = (||\nabla d^T||^2 \frac{\partial^2 b}{\partial d^2})|_{d=10^{-8}l} = 4\times 10^{-16} l^2 \frac{\partial^2 b}{\partial d^2}(10^{-8}l).$$

We then set the lower bound of κ to provide at least 10^{11} times of the average lumped nodal mass $\bar{m}$ on the diagonal entries when $d = 10^{-8}l$ and so enable production of sufficient repulsion at larger distances, that is

$$\kappa_{\min} = 10^{11}\bar{m}/c_{\nabla^2 b}$$

Note that our κ lower bound will be different for different $\hat{d}$, effectly capturing the curvature change of the barrier. With this lower bound, we now can safely use the gradient scheme with bounded κ value.

Distance can still become small if resultant stress or applied compression (e.g, BCs) in the scene are extreme. We thus add an additional, final κ adjustment that doubles κ, when needed, in between Newton iterations. After every Newton iteration, if we detect that there are contact pairs having a characteristic distance smaller than minimum $\hat{d}_\epsilon = 10^{-9}l$ both before and after this iteration, and the distance is decreasing in this Newton iteration, we double the κ value. Although we do not observe divergence of the adapted κ values we apply a fixed upper bound of $\kappa_{\max} = 100\kappa_{\min}$.

To summarize, our adaptive barrier stiffness strategy is:

(1) At start of each time step, compute κ_g giving smallest gradient, and set $\kappa \leftarrow \min(\kappa_{\max}, \max(\kappa_{\min}, \kappa_g))$.

(2) After each Newton iteration, if any contact pair has distance smaller than $\hat{d}_\epsilon$ both before and after this iteration, and the distance is decreasing, set $\kappa \leftarrow \min(\kappa_{\max}, 2\kappa)$.

7 DISTANCE COMPUTATION IMPLEMENTATION

7.1 Point-point and point-edge constraint duplications

As we discuss in our paper many point-triangle and edge-edge distances can and will reduce to point-point or point-edge distance in computation. Thus there can be multiples of exactly the same point-point and/or point-edge stencils in our constraint set. While it is tempting to simply either ignore or remove these duplicates, neither strategy is effective. Ignoring duplication in code can lead to significant redundant computation of the same force and Hessian. On the other hand removing them introduces inconsistency into our objective energy, leading to poor convergence or even divergence over iterations. Instead, we track duplicate stencils, computing their energy, gradient, and Hessian evaluations only once for each distinct stencil and then multiply their entries appropriately so that all terms are correctly applied but still avoiding redundant and expensive computation.

7.2 Nearly parallel edge-edge distance

When computing distances between two nearly parallel edges using the edge-edge plane distance formula (22, main paper), numerical rounding errors will generate huge gradient and Hessian values, and even results in wrong distances (Figure 3) because of the ill-defined normal, making our optimization intractable with double precision floating point numbers. Therefore, we check the angle between edges, forcing each case to reduce to the most appropriate point-point or point-edge constraints if the sine value is smaller than 10^{-10}. This clearly makes the distance function C^0 continuous again at the threshold. However, the nonsmoothness is nearly negligible (Figure 3), while our multiplying energy smoother $e_{k,l}(x)$ is also extremely small when edges are nearly parallel. In practice we find this robustly avoids numerical issues and converges well for all benchmark tests; see Section 7 of our paper.

8 TANGENT AND SLIDING MODES

After reducing the general point-triangle, and edge-edge distances to one of the closed form formulas, see Section 6 in our paper, we can directly compute the sliding basis operators for each of the four types of contact pairs required for computing friction.

We start by defining the basis, $P_k(x) \in R^{3\times 2}$, formed by the two orthogonal 3D unit-length column vectors spanning the tangent space of the contact pair k, and a selection matrix $\Gamma_k \in R^{3\times 3n}$ which computes relative velocity $v_k = \Gamma_k v$ of each contact pair k. Then we can define the sliding basis $T_k(x) = \Gamma_k^T P_k(x)$ that maps tangent space relative velocity or displacement to the stacked global vector. Here we then list the construction for P and Γ for each contact distance type:

Point (x_0) – Triangle ($x_1x_2x_3$).

$$P_k(x) = \left[\frac{x_2-x_1}{||x_2-x_1||}, n\times\frac{x_2-x_1}{||x_2-x_1||}\right] \tag{7}$$

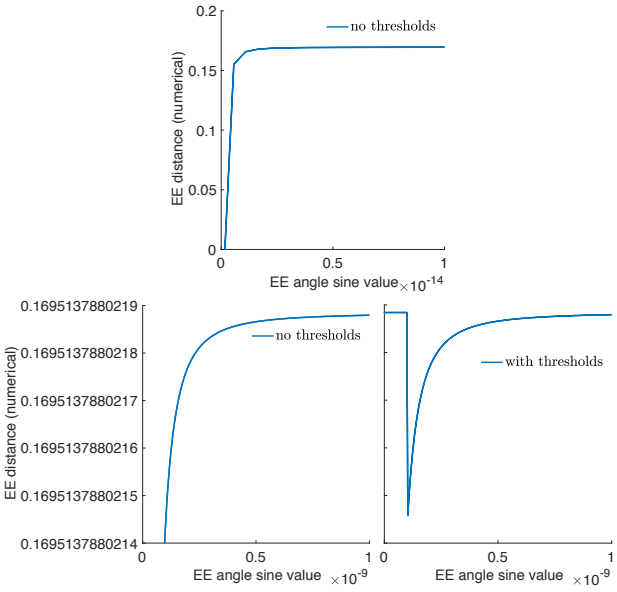

Fig. 3. **Parallel-edge degeneracy handling.** Left: Due to numerical rounding errors for distances of edge-edge pairs decreases to 0 when the edges get more and more parallel (see Figure 7 in our paper for an example); Middle: a zoom-in view show clearly that the distance really starts decreasing when their angle's sine value is at around 10^{-9}; Right: we here set a threshold to force the use of point-point or point-edge formulas to compute the distance of edge-edge pairs when their angle's sine value is below 10^{-10}, which introduces a negligible (for optimization) nonsmoothness.

where $n = \frac{(x_2-x_1)\times(x_3-x_1)}{||(x_2-x_1)\times(x_3-x_1)||}$. Each row of Γ_k is

$$[..., 1, ..., (-1+\beta_1+\beta_2), ..., -\beta_1, ..., -\beta_2, ...] \tag{8}$$

where β's are those defined in $\mathcal{D}^{P-T}$ (17, main paper).

Edge (x_0x_1) – Edge (x_2x_3).

$$P_k(x) = \left[\frac{x_1-x_0}{||x_1-x_0||}, n\times\frac{x_1-x_0}{||x_1-x_0||}\right] \tag{9}$$

where $n = \frac{(x_1-x_0)\times(x_3-x_2)}{||(x_1-x_0)\times(x_3-x_2)||}$. Each row of Γ_k is

$$[..., 1-\gamma_1, ..., \gamma_1, ..., \gamma_2-1, ..., -\gamma_2, ...] \tag{10}$$

where γ's are those defined in $\mathcal{D}^{E-E}$ (18, main paper).

Point (x_0) – Edge (x_1x_2).

$$P_k(x) = \left[\frac{x_2-x_1}{||x_2-x_1||}, \frac{(x_2-x_1)\times(x_0-x_1)}{||(x_2-x_1)\times(x_0-x_1)||}\right] \tag{11}$$

Each row of Γ_k is

$$[..., 1, ..., \eta-1, ..., -\eta, ...] \tag{12}$$

where $(1-\eta)x_1+\eta x_2$ is the closest point to x_0 on edge x_1x_2.

Point x_0 – Point x_1.

$$P_k(x) = \left[t, \frac{x_1-x_0}{||x_1-x_0||}\times t\right] \tag{13}$$

where $t = \frac{e\times(x_1-x_0)}{||e\times(x_1-x_0)||}$ and e is $(1,0,0)$ if (x_1-x_0) is not colinear with $(1,0,0)$, or e is $(0,1,0)$. Each row of Γ_k is $[..., 1, ..., -1, ...]$.

9 FRICTION IMPLEMENTATION

Since we lag the sliding basis in friction computations to $T^n = T(x^n)$ and normal forces to λ^n in (see Section 5 in our paper) from either the last time step or the last friction update iteration n, all other terms are integrable. The lagged friction is then

$$F_k(x,\lambda^n,T^n,\mu) = -\mu\lambda^n T_k^n f_1(||u_k||)\frac{u_k}{||u_k||} \tag{14}$$

and gives us a simple and compact friction potential

$$D_k(x) = \mu\lambda_k^n f_0(||u_k||). \tag{15}$$

Here f_0 is given by $f_0' = f_1$ and $f_0(\epsilon_v h) = \epsilon_v h$ so that $F_k(x) = -\nabla D_k(x)$. In turn this likewise provides a simple-to-compute Hessian contribution

$$\begin{aligned}\nabla^2 D_k(x) = \mu\lambda_k^n T_k^n\Big(&\frac{f_1'(||u_k||)||u_k|| - f_1(||u_k||)}{||u_k||^3}u_k u_k^T \\ &+ \frac{f_1(||u_k||)}{||u_k||}I_2\Big)T_k^{nT}.\end{aligned} \tag{16}$$

where $I_2 = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$. Projecting this Hessian to PSD then simply requires projecting the 2×2 matrix

$$\frac{f_1'(||u_k||)||u_k|| - f_1(||u_k||)}{||u_k||^3}u_k u_k^T + \frac{f_1(||u_k||)}{||u_k||}I_2 \tag{17}$$

to SPD as T_k^n is symmetrically multiplied on both of its two sides.

The above model is general so that f_0 and f_1' are both easy to define for a range of f_1 choices:

(1) C^0 F_f: $f_0(x) = \frac{x^2}{2\epsilon_v h} + \frac{\epsilon_v h}{2}$, $f_1(x) = \frac{x}{\epsilon_v h}$, and $f_1'(x) = \frac{1}{\epsilon_v h}$;
(2) C^1 F_f: $f_0(x) = -\frac{x^3}{3\epsilon_v^2 h^2} + \frac{x^2}{\epsilon_v h} + \frac{\epsilon_v h}{3}$, $f_1(x) = -\frac{x^2}{\epsilon_v^2 h^2} + \frac{2x}{\epsilon_v h}$, and $f_1'(x) = -\frac{2x}{\epsilon_v^2 h^2} + \frac{2}{\epsilon_v h}$;
(3) C^2 F_f: $f_0(x) = \frac{x^4}{4\epsilon_v^3 h^3} - \frac{x^3}{\epsilon_v^2 h^2} + \frac{3x^2}{2\epsilon_v h} + \frac{\epsilon_v h}{4}$, $f_1(x) = \frac{x^3}{\epsilon_v^3 h^3} - \frac{3x^2}{\epsilon_v^2 h^2} + \frac{3x}{\epsilon_v h}$, and $f_1'(x) = \frac{3x^2}{\epsilon_v^3 h^3} - \frac{6x}{\epsilon_v^2 h^2} + \frac{3}{\epsilon_v h}$.

Importantly this also emphasizes that there are never any divisions by $||u_k||$ in all of our energy, gradient, and Hessian computations for friction as they are always cancelled out. This ensures that so that the implemented computation can be robust and accurate.

Our friction model applies our C^1 – (2) in the above.. This design choice again provides a continuous Hessian for better convergence in our Newton-type method. Here we also provide a comparison of behavior of the C^1 model w.r.t. to the different orders of smoothed friction model on our arch and ball roller example (Figure 4).

Examples	C0 # iters, t (s) / time step	C1 (IPC) # iters, t (s) / time step	C2 # iters, t (s) / time step
sphere1K roller	1.37, 0.01	1.24, 0.01	1.26, 0.02
1m-height arch (static)	53.60, 12.21	45.22, 9.79	52.83, 11.42

Fig. 4. **Ablation study on smoothed static friction.**

We observe that C^1 friction model provides a "sweet-spot": improvement over C^0 due to the C^1 model's continuous hessian, while it also improves over the C^2 model with less additional nonlinearity.

10 SQUARED TERMS

In our implementation, we apply squared distances in our evaluations to avoid numerical errors and inefficiencies that can be introduced by taking squared roots – especially in gradient and Hessian computations. Concretely, our barrier terms are applied as $b(d^2, \hat{d}^2)$ throughout our implementation. This manipulation leaves our problem formulation with unsigned distances unchanged as have $d > 0 \Leftrightarrow d^2 > 0$. However, we must be careful with units in order to preserve appropriate scaling of dimensions – especially in terms of friction. Fortunately, we can sort this out directly via the chain rule.

To ensure that units of normal force remain correct we now observe that directly plugging in d^2 and $\hat{d}^2$ into Equation (9) in our paper no longer computes the contact force magnitudes λ_k defined in our paper. Here we now have $-\frac{\kappa}{h^2}\frac{\partial b}{\partial (d_k^2)}(d_k^2, \hat{d}_k^2)$. Applying the squared formulation we rewrite the stationarity of the barrier as

$$M(\frac{x - \hat{x}}{h^2}) = -\nabla\Psi(x) - \frac{\kappa}{h^2}\sum_{k\in C}\frac{\partial b}{\partial (d_k^2)}\frac{\partial (d_k^2)}{\partial (d_k)}\nabla d_k(x) \tag{18}$$

which is $M(\frac{x-\hat{x}}{h^2}) = -\nabla\Psi(x) - \frac{\kappa}{h^2}\sum_{k\in C}\frac{\partial b}{\partial (d_k^2)}2d_k\nabla d_k(x)$. In turn this allows us to extract the correct contact force magnitudes (when using squared distances) as $\lambda_k = -\frac{\kappa}{h^2}\frac{\partial b}{\partial (d_k^2)}2d_k$.

REFERENCES

Jorge Nocedal and Stephen Wright. 2006. *Numerical optimization.* Springer Science & Business Media.