Medial IPC: Accelerated Incremental Potential Contact with Medial Elastics

LEI LAN*, Clemson University, USA & Xiamen University, China
YIN YANG*, Clemson University, USA
DANNY KAUFMAN, Adobe Research, USA
JUNFENG YAO, Xiamen University, China
MINCHEN LI, UCLA, University of Pennsylvania, USA
CHENFANFU JIANG, UCLA, University of Pennsylvania, USA

Fig. 1. Medial IPC is a versatile reduced simulation framework that unifies nonlinear hyperelastic simulation and continuous self-/collision detection with medial axis transform. It inherits the superior capability of shape approximation of medial mesh and the strong robustness of continuous collision resolve of IPC. As a result, medial IPC produces high-quality deformable animations with rich and intensive collisions and contacts but at a significantly accelerated rate. In the experiment shown in the figure, two barbarian ships falls on an array of thin rods. Each ship has nearly half million elements, and they frequently interact with each other during the falling. There are many fine elastic paddles at both sides of the ship, multiple ladders, masts and canvases. Interpenetrations among those small geometries can be easily generated if discrete collision detection is used (i.e., see the highlights in the figure). We also aggressively set the time step size to 1/30. In this challenging simulation, Medial IPC robustly handles all the collision and contact events within a subspace of 16, 272 dimension and yields compelling animation results. In this experiment, medial IPC is 110× faster than the fullspace IPC simulation.

We propose a framework of efficient nonlinear deformable simulation with both fast continuous collision detection and robust collision resolution. We name this new framework *Medial IPC* as it integrates the merits from medial elastics, for an efficient and versatile reduced simulation, as well as incremental potential contact, for a robust collision and contact resolution. We leverage medial axis transform to construct a kinematic subspace. Instead of resorting to projective dynamics, we use classic hyperelastics to embrace real-world nonlinear materials. A novel reduced continuous collision detection algorithm is presented based on the medial mesh. Thanks to unique geometric properties of medial axis and medial primitives, we derive closed-form formulations for identifying between-primitive collision within the reduced medial space. In the meantime, the implicit barrier energy that generates necessary repulsion forces for collision resolution is also formulated with the medial coordinate. In other words, Medial IPC exploits a universal reduced coordinate for simulation, continuous self-/collision detection, and IPC-based collision resolution. Continuous collision detection also allows more aggressive time stepping. In addition, we carefully implement our system with a heterogeneous CPU-GPU deployment such that massively parallelizable computations are carried out on the GPU while few sequential computations are on the CPU. Such implementation also frees us from generating training poses for selecting Cubature points and pre-computing their weights. We have tested our method on complicated deformable models and collision-rich simulation scenarios. Due to the reduced nature of our system, the computation is faster than fullspace IPC or other fullspace methods using continuous collision detection by at least one order. The simulation remains high-quality as the medial subspace captures intriguing and local deformations with sufficient realism.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; **Collision detection**.

Additional Key Words and Phrases: Medial axis, Continuous collision detection, GPU, Deformable model

ACM Reference Format:
Lei Lan, Yin Yang, Danny Kaufman, Junfeng Yao, Minchen Li, and Chenfanfu Jiang. 2021. Medial IPC: Accelerated Incremental Potential Contact with Medial Elastics. *ACM Trans. Graph.* 40, 4, Article 158 (August 2021), 16 pages. https://doi.org/10.1145/3450626.3459753

*Joint first authors

Authors' addresses: Lei Lan, Clemson University, USA & Xiamen University, China, lanlei.virhum@gmail.com; Yin Yang, Clemson University, USA, yin5@clemson.edu; Danny Kaufman, Adobe Research, USA, danny.kaufman.cs@gmail.com; Junfeng Yao, Xiamen University, China, yao0010@xmu.edu.cn; Minchen Li, UCLA, University of Pennsylvania, USA, minchernl@gmail.com; Chenfanfu Jiang, UCLA, University of Pennsylvania, USA.

0730-0301/2021/8-ART158 $15.00
https://doi.org/10.1145/3450626.3459753

1 INTRODUCTION

Deformable simulation has been extensively studied in the graphics community. While the underlying computation methods are well established, the computational cost for simulating high-resolution geometrically complex deformable objects remains prohibitive for applications with a limited time budget. It is common to use coarsened or reduced simulation techniques to formulate elastic dynamics with a generalized coordinate [Sifakis and Barbic 2012]. By doing so, the core simulation becomes independent of the size of the input model. During the simulation, it is also expected that deformable objects do not overlap or interpenetrate with each other. This requirement is enforced by a dedicated self-/collision detection routine which returns all intersecting triangle pairs. Given this information, we could resolve the collision by using complimentary constraints or penalty forces [Baraff 1994; Moore and Wilhelms 1988].

In dynamic simulation, we discretize the temporal domain into time steps for the ease of time differentiation. It is reasonable to perform collision detection by the end of a time step and resolve detected collisions in the next step. This strategy is known as the discrete collision detection (DCD). While implementation-friendly, DCD runs into several robustness and accuracy concerns in practice especially under bigger time steps (e.g., 1/30 sec). For instance, a fast-moving vertex could easily pass through small-scale geometries in one step. DCD is inherently limited for such single-step penetrations, not only producing annoying visual artifacts but also leading to severe physical inaccuracy. In contrast, continuous collision detection (CCD) [Brochu et al. 2012] offers a more robust solution. Unlike DCD that queries collisions at a given time instance, CCD scrutinizes potential overlaps of linearized triangle *trajectories* over the entire time step, which in turn is more computationally involved.

Several research has demonstrated convincing advantages combining reduced simulation and collision detection. For instance, James and Pi [2004] exploited subspace modes to construct a bounded deformation tree (BD-tree) for fast collision culling. Barbič and James [2010] accelerated self-collision culling by pre-computing subspace certificates, which demarcate a self-collision free area on the model with reduced coordinates. Recently, Lan and colleagues [Lan et al. 2020] leveraged medial axis transform (MAT) to accelerate both projective dynamics (PD) simulation and culling. We note that however, the benefits brought by model reduction mainly contribute to DCD culling in existing methods. Sophisticated computations for CCD are largely untouched and remain in the fullspace.

In this paper, we stick with the high-level concept of leveraging complementary features from reduced simulation and collision detection. Apart from previous contributions however, we focus on an algorithmic integration of continuous collision detection and handling. We note that CCD always synergizes with implicit collision resolve algorithms. They two-way couple all the colliding objects making the system solve much more expensive, no to mention extra numerical safeguards needed for trimming oversized Newton step. From this perspective, we believe model reduction ought to be more effective and desired in CCD-involved simulation problems. Driven by this idea, we propose a novel CCD-capable reduced simulation framework. We name our system *Medial IPC* as it inherits favored merits from two recent contributions of medial elastics [Lan et al. 2020] and incremental potential contact (IPC) [Li et al. 2020] in order to reach a better trade-off between accuracy and performance. With medial IPC, we substantially accelerate full IPC simulation, and the resulting animation is highly plausible and realistic. While conceptually straightforward, our system addresses several significant technical challenges in model reduction and CCD:

- From the model reduction point of view, we build our reduced kinematics with MAT, similar to medial elastics [Lan et al. 2020]. MAT is known as the topology skeleton, and it intrinsically captures nonlinear shape deformation of a 3D model. The deformation effects are rich and vivid with just few hundred handles. Our system offers general hyperelastic simulation instead of using PD. All collision and contact events are handled robustly in a full implicit manner.
- Our framework is CCD friendly and efficient: all the CCD computations are directly based on the generalized degrees of freedom (DOFs) at medial handles, thus independent of the surface resolutions. We give the detailed formulation for calculating the first time contact of colliding medial primitives. This is a hexic polynomial function and solved numerically. Thanks to model reduction, we are able to exhaustively run CCD tests efficiently on GPU (i.e., in milliseconds) making collision culling unnecessary.
- We follow the paradigm of IPC [Li et al. 2020] to resolve collision and contact instances. Fullspace IPC plugs a potential energy when triangles move sufficiently close to each other and yields an increasingly stronger repulsion force to keep them apart. In our system, this energy is also natively defined in the reduced space. Without fullspace-subspace conversion, its differentiation directly gives generalized collision force to forward the simulation. Finally, we adopt a hybrid CPU-GPU implementation that fully harvests the power of hardware and pushes the simulation performance to the limit.

In the experiments, we find that medial IPC runs consistently faster than fullspace IPC by at least one order. With such substantially accelerated simulation, the resulting animation retains a very high quality. An example is reported in Fig. 1, where two barbarian ships are falling, colliding, and interacting with the environment actively during the simulation. Our MAT-based model reduction forms a subspace of 16,272 dimensions. All the computations for CCD, IPC, and Neo-Hookean dynamics are carried out in the subspace. This makes the simulation over 110× faster than the fullspace IPC simulation. While the scale of the simulation is condensed by 50×, medial IPC manages to capture a wide range of deformations with little visible difference. The residual errors stay below 1% in both cases with all collisions resolved at the time of the contact.

2 RELATED WORK

Deformable simulation solves a dynamic equilibrium among the external force, inertia force, and the nonlinear internal force. For large-scale models i.e., with hundred thousands of DOFs, the simulator needs to solve a high-dimension nonlinear system repetitively at each time step. Therefore, even there are many well-established simulation frameworks such as finite element method (FEM) [Zienkiewicz et al. 1977], finite difference method [Zhu et al. 2010], meshless method [Martin et al. 2010; Müller et al. 2005],

mass-spring system [Liu et al. 2013] etc, efficient simulating high-resolution models is still a challenging problem.

Speeding up a deformable simulation can be achieved using dedicated numerical treatments like the multigrid method [Tamstorf et al. 2015; Zhu et al. 2010], an incremental matrix update [Hecht et al. 2012], or parallelizable solvers [Fratarcangeli et al. 2016; Wang and Yang 2016]. These methods focus on improving the performance for the fullspace nonlinear optimization without condensing the simulation scale. Acceleration can also be achieved using *model reduction*, which removes less important DOFs and creates a subspace representation of fullspace DOFs. Proper Orthogonal Decomposition (POD) [Barbič and Popović 2008], modal analysis [Choi and Ko 2005; Hauser et al. 2003; Pentland and Williams 1989] and its first-order derivatives [Barbič and James 2005; Yang et al. 2015] are often considered as highly effective approaches for the subspace construction. Displacement vectors from recent fullspace simulations can also be utilized as subspace bases [Kim and James 2009]. Alternatively, it is also viable to coarsen geometric shape representation to prescribe the dynamics of a fine model like skin rigging widely used in modern animation systems. Analogously, Capell and colleagues [2002] deformed an elastic body using an embedded skeleton; Gilles and colleagues [2011] used 6-DOF rigid frames to drive the deformable simulation; Faure and colleagues [2011] used scattered *handles* to model complex deformable models; Martin and colleagues [2010] used sparsely-distributed integrators named *elastons* to model the nonlinear dynamics of rod, shell, and solid uniformly.

Collision detection is another important task along the physics animation pipeline. In theory, collisions could occur at any surface triangle pairs, and an exhaustive triangle-based collision detection is infeasible for high-resolution models. To this end, a commonly adopted method is to use some bounding volume hierarchy (BVH) [Zachmann and Langetepe 2003] to avoid excessive triangle-triangle intersection tests. This pre-screening procedure is also known as collision culling. Different BV types have been explored such as AABB [Bergen 1997], OBB [Gottschalk et al. 1996], bounding sphere [Hubbard 1995; James and Pai 2004], Boxtree [Zachmann 2002], spherical shell [Krishnan et al. 1998] and so on. It is noted that both model reduction and collision culling seek for good shape approximations. Following this intuition, James and Pi [2004] proposed an algorithm that updates the BD-tree directly using the generalized coordinate. Barbič and James [2010] computed self-collision *certificates* in subspace to accelerate self-collision culling. Following the observation that a self-collision occurs under large local deformation, Zheng and James [2012] proposed an energy-based metric to improve the culling effectiveness. Recently, Lan and colleagues designed a MAT-based reduced simulator called medial elastics [Lan et al. 2020]. As the MAT fits the model geometry much tighter than other BVs, collision culling becomes more effective.

Nevertheless, the integration of model reduction and CCD at algorithmic level remains challenging. This is because after culling, the actual triangle-triangle intersection tests still occur in the fullspace using per-vertex positions [Ainsley et al. 2012; Harmon et al. 2008; Otaduy et al. 2009]. In CCD, one also needs to calculate the first time contact, which is not well-defined in a general-purpose subspace (such as modal spaces). CCD should also be paired with implicit collision resolve algorithms [Geilinger et al. 2020; Macklin et al. 2020, 2019]. Few methods offer guaranteed convergence and accuracy even in the fullspace. Naïve fullspace-subspace projection of collision forces also induces accuracy loss and potentially undermines the effectiveness of the collision force calculated in fullspace. Therefore, accelerating CCD-based nonlinear elasticity with high robustness and accuracy is an unsolved problem.

Medial IPC is our attempt towards this challenge. Our method is inspired by two recent contributions in model reduction and implicit collision/contact resolve. Specifically, our reduced dynamics follows a similar form in medial elastics [Lan et al. 2020], by prescribing vertex kinematics based on deformation handles placed at medial vertices of MAT [Blum 1967]. MAT essentially provides a tight volume enclosure of the input model, which greatly facilitates collision culling in medial elastics – a few hundred medial primitives are able to tightly encapsulate complicated models. This pleasing feature is better utilized in our framework, as we completely avoid triangle-level computations for collision, self-collision, and contact. This is achieved by designing the barrier function of the incremental potential within the medial space. In other words, our framework monitors the closest unsigned distance between each pair of medial primitives, and an IPC penalty [Li et al. 2020] rises yielding collision and friction forces needed whenever a barrier function is activated. We enable most numerical treatments developed by fullspace IPC, including such as positive-semi definite projection of the element stiffness matrix and integrating per-iteration CCD in line search despite moving all of them to a subspace formulation. We would like to remind that such subspace re-formulation is not merely projecting their fullspace counterparts into the subspace, which could bring accuracy loss. Instead, our subspace formulation is *native*. This is not a trivial task. As medial primitives can be considered as an interpolated sphere surface, computing the first time of contact for MAT results in a complicated higher-order problem. We manage to give its analytic formulation, and solve this equation numerically.

Implementation also imposes many challenges. Usually, general model reductions aim to keep all simulation algorithms at the order of the subspace size n (e.g., n denotes the total number of medial handles in our framework). Therefore, they rely on Cubature sampling [An et al. 2008] to avoid computation at $\mathbf{O}(N)$, where N stands for the size of the input model. In our framework, we fully leverage GPU to speed up the reduced force and Hessian computation. A tricky problem is, when we have a large-scale model (e.g., with hundred thousands of vertices) and a big subspace (e.g., of several thousand dimension), we may not have enough GPU memory storing the subspace matrix, whose space complexity is $\mathbf{O}(N \cdot n)$. To this end, we exploit unique algebraic structure of medial subspace so that this operation could be processed with consumer-level GPUs.

3 REDUCED SIMULATION WITH MEDIAL ELASTICS

The fullspace deformable simulation is formulated as: $\mathbf{M\ddot{u}} = \mathbf{f}_{int}(\mathbf{u}) + \mathbf{f}_{ext}$ implying the equilibrium among inertia force ($\mathbf{M\ddot{u}}$), internal force ($\mathbf{f}_{int}$), and external force ($\mathbf{f}_{ext}$). While $\mathbf{f}_{ext}$ is often considered known, $\mathbf{f}_{int}$ could be nonlinearly dependent on the unknown displacement of the system ($\mathbf{u}$). Model reduction assumes a linear subspace is able to well express the fullspace displacement such as $\mathbf{u} = \mathbf{Uq}$, where $\mathbf{U} \in \mathbb{R}^{N \times n}$ is a slim subspace matrix (i.e., $n << N$).

This assumption allows us to project the original fullspace equilibrium into the n-dimension column space of $\mathbf{U}$:

$$\widetilde{\mathbf{M}}\ddot{\mathbf{q}} = \widetilde{\mathbf{f}}_{int}(\mathbf{q}) + \mathbf{U}^\top \mathbf{f}_{ext}. \tag{1}$$

$\widetilde{\mathbf{M}} = \mathbf{U}^\top \mathbf{M}\mathbf{U}$ and $\widetilde{\mathbf{f}}_{int}$ are the reduced mass matrix and internal force.

For a given 3D shape, there exists a unique medial axis transform or MAT that *losslessly* encodes its surface geometry. The MAT consists of a set of spheres. Each sphere is maximally inscribed with at least two closest points on the boundary of the model [Blum 1967]. With MAT, one can easily perform the in/out test by comparing the distance to the nearest medial point and the corresponding sphere radius. MAT can be aggressively simplified to a *medial mesh*, a 2D non-manifold mesh of triangles and edges [Sun et al. 2016]. The radius information at medial vertices is interpolated along edges and triangles forming the so-called medial primitives (or simply primitive in this paper), which is either a cone or a slab (see Fig. 2).

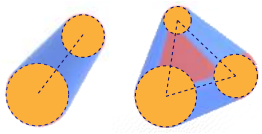

Fig. 2. A medal primitive is either medial cone (left) or a medial slab (right).

Medial elastics prescribes kinematics in a skinning-like manner: the deformation of the model is driven by the handles at each medial vertex. A handle holds deformation DOFs i.e., a transformation stencil, and the final deformation is computed by blending deformation from all the handles. In medial IPC, we assign each handle an affine transformation as did in [Faure et al. 2011]. Given the rest position $\mathbf{x}_i$ of the vertex i, its displacement $\mathbf{u}_i$ is written as:

$$\mathbf{u}_i = \mathbf{U}\mathbf{q} = [w_1\mathbf{U}(\mathbf{x}_i), \cdots, w_n\mathbf{U}(\mathbf{x}_i)] \begin{bmatrix} \mathbf{q}^1 \\ \vdots \\ \mathbf{q}^n \end{bmatrix}, \tag{2}$$

where w_j from the j-th handle is the biharmonic weight [Jacobson et al. 2011]. We *do not* add high-order DOFs as in other spatial reduction simulators [Bargteil and Cohen 2014; Luo et al. 2018; Martin et al. 2010]. Consequently, the subspace matrix holds a simple and relatively sparse pattern of:

$$\mathbf{U}(\mathbf{x}_i) = \left[\mathbf{I}, \mathbf{I} \otimes \mathbf{x}_i^\top\right] \in \mathbb{R}^{3\times 12}. \tag{3}$$

If we split the generalized coordinate $\mathbf{q}$ into four three-vectors such that $\mathbf{q} = [\mathbf{q}_1^\top, \mathbf{q}_2^\top, \mathbf{q}_3^\top, \mathbf{q}_4^\top]^\top$, it can be easily verified that $\mathbf{U}(\mathbf{x}_i)\mathbf{q}$ is essentially equivalent to $\mathbf{A}\mathbf{x}_i + \mathbf{q}_1$ with $\mathbf{A} = [\mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4]$. If quadratic or other nonlinear DOFs [Gilles et al. 2011] are involved, the structure of the subspace matrix becomes complicated and dense, which overstresses the GPU memory footprint and prohibits a Cubature-free subspace integration (e.g., see § 6). As long as the sparsity of the handles are moderate (e.g., a few hundred), we do not observe any visual artifacts in our experiments even under large deformations.

Another difference between our work and medial elastics is the solver choice. We do not use a PD-based solver [Bouaziz et al. 2014; Brandt et al. 2018]. Instead, medial IPC sticks with classic Lagrangian mechanics. In our implementation, we choose a variant of the Neo-Hookean material model proposed in [Smith et al. 2018], which possesses an altered positive semi-definite (PSD) Hessian and a strong volume preserving penalty:

$$\Psi(\mathbf{F}) = \frac{\mu}{2}(I_C - 3) + \frac{\lambda}{2}(J - \alpha)^2 - \frac{\mu}{2}\log(I_C + 1). \tag{4}$$

Here, $J = |\mathbf{F}|$; $I_C = \mathrm{tr}(\mathbf{F}^\top\mathbf{F})$; $\mathbf{F}$ is the deformation gradient tensor; and $\alpha = 1+\mu/\lambda-\mu/(4\lambda)$ is a rest-stability shift. μ and λ are Lamé constants of the material. More detailed derivation of this energy can be found in related literature e.g., see [Smith et al. 2018] and [Kim and Eberle 2020]. Implementing other material models are straightforward with our system and should not be claimed as a contribution. Our reduced model maintains a subspace of a considerable dimension, which permits sharp and local deformations. Numerical treatments like PSD projection and principle stress correction [Irving et al. 2004; Xu et al. 2015] will be needed. Such numerical hazards however, seldom occur when an aggressive reduction is applied for instance, as in [Barbič and James 2005].

The internal force $\mathbf{f}_{int}$ is resultant of elastic resistance, collision and contact, and damping. It follows the negative gradients of those energies and is linearized at each Newton step. The collision and contact follow the IPC model [Li et al. 2020] but directly based on the generalized coordinate $\mathbf{q}$. As we will see later, this strategy reinforces the advantage of model reduction as the collision and contact forces are natively in generalized form without the need for extra fullspace-subspace projections, which are often the computation bottleneck in model reduction.

4 MEDIAL DISTANCE & MEDIAL IPC ENERGY

Our rationale, when aiming an CCD-aware efficient simulation, is that the reduced basis should hold an articulated geometric information, which could be fully exploited in the CCD procedure. Hence even modal analysis [Pentland and Williams 1989] provides a theoretically optimal subspace (around the rest shape), it is not adopted in our framework. After all, when collisions are intensive, high-frequency deformations abound, and a low-frequency-dominant subspace becomes potentially problematic. MAT-based model reduction on the other hand, excels in this regard, which allows us to formulate a *medial continuous collision detection* algorithm without referring to fullspace per-vertex position or displacement (see Fig. 3). It should be noted that the reduced representations for model deformation and CCD are not consistent. That is the deformation of medial primitives does not perfectly align with the vertices. Therefore, medial IPC essentially uses an approximated CCD processing over high-resolution meshes. We also assume there is no triangle self-collision within a primitive. As the total number of primitives is limited, we can simply perform a brute-force CCD for all primitive pairs without needing collision culling

Fig. 3. MAT with 127 primitives (left) well approximates the geometry of the dinosaur (right).

4.1 Medial Distance

In medial IPC, we need to compute the closest distance between any pair of medial primitives. If this distance is smaller than a pre-defined *barrier activation threshold* $\widehat{d} > 0$, the barrier function becomes non-zero, resulting in collision and friction forces. In other words, the collision force is generated before two objects physically touch with

each other, and $\widehat{d}$ determines the smoothness of a collision event. By default, $\widehat{d}$ is defined based on fullspace vertex positions [Li et al. 2020]. In medial IPC, it is alternatively defined with an altered distance metric between two medial primitives named *medial distance*. Medial distance needs to be re-calculated every time after $\mathbf{q}$ is updated. It is also noteworthy that collision culling is *not* practiced in medial IPC. As all the computations are fully inside the medial subspace, distance (and other related) calculations are light-weight and easily parallelizable on GPU.

Without loss of the generality, we elaborate our formulation for the cone-cone medial distance. Sphere-slab distance follows a similar procedure (details are in the supplementary document). They together allow us to compute slab-cone and slab-slab cases. For instance, a slab-cone test consists of one sphere-slab test and three cone-cone tests. Let $c_{1,1}$ and $c_{1,2}$ denote sphere centers at two ends of C_1, whose radii are $r_{1,1}$ and $r_{1,2}$. Likewise, $c_{2,1}$, $c_{2,2}$ and $r_{2,1}$, $r_{2,2}$ represent sphere centers and radii of C_2. Any inscribing sphere in C_1 and C_2 can then be interpolated as:

$$\begin{aligned} c_1(\alpha) &= \alpha(c_{1,1} - c_{1,2}) + c_{1,2}, \quad r_1(\alpha) = \alpha(r_{1,1} - r_{1,2}) + r_{1,2}, \\ c_2(\beta) &= \beta(c_{2,1} - c_{2,2}) + c_{2,2}, \quad r_2(\beta) = \beta(r_{2,1} - r_{2,2}) + r_{2,2}, \end{aligned}$$

where $\alpha, \beta \in [0, 1]$ are interpolation parameters in C_1 and C_2.

When C_1 and C_2 stay apart, their closest distance can be written as a minimization problem:

$$\min_{\alpha,\beta} \|c_1(\alpha) - c_2(\beta)\| - (r_1(\alpha) + r_2(\beta)). \tag{5}$$

Eq. (5) computes the closest Euclidean distance between C_1 and C_2 by subtracting the distance between corresponding sphere centers by their radii. We found that however, because of the square rooting term in $\|c_1(\alpha) - c_2(\beta)\|$, formulating an IPC energy and its derivative based on Eq. (5) becomes quite complicated. As this is used in our IPC formulation as a barrier to prevent C_1 and C_2 from moving towards each other, we construct medial distance by a slight modification of Eq. (5) as:

$$\min_{\alpha,\beta} f, \quad f = \|c_1(\alpha) - c_2(\beta)\|^2 - (r_1(\alpha) + r_2(\beta))^2. \tag{6}$$

We note that the medial distance f is zero if and only if C_1 is in contact with C_2. It monotonously increases, as the Euclidean distance does, when C_1 and C_2 move away from each other (but at a different rate). Those observations agree that an IPC energy based on Eq. (6) will be as viable as the one forged with Eq. (5) in terms of pushing the colliding cones away. Lastly, we would like to remind that neither of Eqs. (5) or (6) works when C_1 intersects with C_2. C_1 and C_2 being separated by a non-negative distance is the prerequisite of not only the IPC method but also of interior-point-based numerical procedures [Den Hertog 2012].

Expanding the medial distance formulation in Eq. (6) with some manipulations could better reveal the structure of this metric:

$$\begin{aligned} f(\alpha, \beta) &= \|\alpha C_1 + \beta C_2 + C_3\|^2 - (\alpha R_1 + \beta R_2 + R_3)^2 \\ &= A\alpha^2 + B\alpha\beta + C\beta^2 + D\alpha + E\beta + F, \end{aligned} \tag{7}$$

where coefficients of this quadratic equation are defined as:

$$\begin{aligned} C_1 &= c_{1,1} - c_{1,2}, \quad C_2 = -c_{2,1} - c_{2,2}, \quad C_3 = c_{1,2} - c_{2,2}, \\ R_1 &= r_{1,1} - r_{1,2}, \quad R_2 = r_{2,1} - r_{2,2}, \quad R_3 = r_{1,2} + r_{2,2}, \end{aligned} \tag{8}$$

and

$$\begin{aligned} &A = C_1^\top C_1 - R_1^2, \quad B = 2(C_1^\top C_2 - R_1 R_2), \quad C = C_2^\top C_2 - R_2^2, \\ &D = 2(C_1^\top C_3 - R_1 R_3), \quad E = 2(C_2^\top C_3 - R_2 R_3), \quad F = C_3^\top C_3 - R_3^2. \end{aligned} \tag{9}$$

In order to determine should a medial IPC energy be plugged for C_1 and C_2, we need to solve for interpolation parameters α and β in Eq. (7). The minimizer can be calculated by setting the partial derivatives of $\partial f/\partial\alpha = 0$ and $\partial f/\partial\beta = 0$ as:

$$\alpha^* = \frac{BE - 2CD}{4AC - B^2}, \quad \beta^* = \frac{BD - 2AE}{4AC - B^2}. \tag{10}$$

If $f^* = f(\alpha^*, \beta^*) \le \widehat{d}$, the barrier function activates, and a collision or contact force appears in $\widetilde{\mathbf{f}}_{int}$ (Eq. (1)). In this case, we need to further calculate the gradient and Hessian of the medial IPC energy.

It may be immediately noticed that Eq. (10) is not numerically stable and could lead to the division-by-zero issue in practice. In addition, Eq. (10) gives unconstrained optimal values of α and β as the resulting minimizer α^* and β^* may fall outside of the designated interval of $[0, 1]$. In the following subsection, we expatiate how to compute corresponding IPC gradient and Hessian to avoid those abnormalities.

4.2 Gradient and Hessian of Medial IPC

In medial IPC, we use the same the barrier equation as in the original IPC [Li et al. 2020] except with the medial distance (Eq. (6)):

$$b(f, \widehat{d}) = \begin{cases} -(f - \widehat{d})^2 \ln\left(\dfrac{f}{\widehat{d}}\right) & , \ 0 < f \le \widehat{d} \\ 0 & , \ f > \widehat{d} \end{cases}. \tag{11}$$

Its gradient and Hessian can be captured by applying the chain rule as:

$$\frac{\partial b}{\partial \mathbf{q}} = \frac{\partial b}{\partial f}\frac{\partial f}{\partial \mathbf{q}}, \quad \frac{\partial^2 b}{\partial \mathbf{q}^2} = \left(\frac{\partial f}{\partial \mathbf{q}}\right)^\top \frac{\partial^2 b}{\partial f^2}\frac{\partial f}{\partial \mathbf{q}} + \frac{\partial b}{\partial f}\frac{\partial^2 f}{\partial \mathbf{q}^2}. \tag{12}$$

As an elementary function, $\partial b/\partial f$ and $\partial^2 b/\partial f^2$ are trivial to obtain. The first- and second-order partial derivatives of the medial distance on the other hand, undergo another chain rule:

$$\begin{aligned} \left.\frac{\partial f}{\partial \mathbf{q}}\right|_{f=f(\alpha^*,\beta^*)} &= \sum \frac{\partial f}{\partial X}\frac{\partial X}{\partial \mathbf{q}}, \\ \left.\frac{\partial^2 f}{\partial \mathbf{q}^2}\right|_{f=f(\alpha^*,\beta^*)} &= \sum \frac{\partial^2 f}{\partial X^2}\left(\frac{\partial X}{\partial \mathbf{q}}\right)^\top \frac{\partial X}{\partial \mathbf{q}} + \sum \frac{\partial f}{\partial X}\frac{\partial^2 X}{\partial \mathbf{q}^2}, \end{aligned} \tag{13}$$

where X stands for coefficients A to F in Eq. (9). Their derivatives with respect to $\mathbf{q}$ are given in Appendix A. It may appear confusing as Eq. (13) does not contain any partial derivatives of α and β. This is because $\partial f/\partial \mathbf{q}$ and $\partial^2 f/\partial \mathbf{q}^2$ are evaluated at $f = f^*$, where $\partial f/\partial\alpha$, $\partial^2 f/\partial\alpha^2$ and $\partial f/\partial\beta$, $\partial^2 f/\partial\alpha^2$ all have vanished values if α and β are the global minimizer of f. There are however some corner cases need to be taken care of.

■ **Case 1**: $4AC - B^2 = 0$.
As shown in Fig. 4, $4AC - B^2 = 0$ indicates edges of C_1 and C_2 are in parallel, and the radii of medial spheres at each cone are invariant. In **Case 1**, it is possible that solutions of α^* and β^* are not unique, and Eq. (10) is no longer applicable. To compute medial distance, we first examine the smallest value among $f(0, 0)$, $f(0, 1)$, $f(1, 0)$,

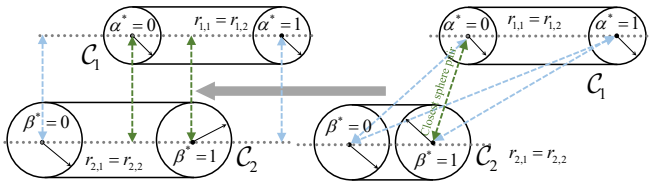

Fig. 4. **Case 1**: two parallel cones with fixed radii make $4AC-B^2=0$. When two edge spans have an overlap, α^* and β^* are not unique (left). Otherwise, α^* and β^* should be either 0 or 1 (right).

and $f(1,1)$, which correspond to four arrows in Fig. 4 (right). For the closest sphere pair, we calculate the medial distance at one of its sphere center. For the example of Fig. 4, we can solve for β^* using Eq. (7) by setting $\alpha^*=0$. Alternatively, we can also solve for α^* while fixing $\beta^*=1$. They correspond to calculating one of the two green arrows in Fig. 4 left. However, this computation only needs to be performed once, either $\alpha^*=0$ or $\beta^*=1$.

Assume we let $\alpha^*=0$. If $\beta^*\in[0,1]$, $f(0,\beta^*)$ is the medial distance between C_1 and C_2, and the computation of the distance gradient/Hessian follows the procedure described in **Case 2** or **Case 3**. Otherwise, the medial distance should be computed based on the closest sphere centers at cone ends. If so, the reduced gradient and Hessian are:

$$\begin{aligned}\frac{\partial f}{\partial \mathbf{q}} &= \alpha^2\frac{\partial A}{\partial \mathbf{q}}+\alpha\beta\frac{\partial B}{\partial \mathbf{q}}+\beta^2\frac{\partial C}{\partial \mathbf{q}}+\alpha\frac{\partial D}{\partial \mathbf{q}}+\beta\frac{\partial E}{\partial \mathbf{q}}+\frac{\partial F}{\partial \mathbf{q}},\\ \frac{\partial^2 f}{\partial \mathbf{q}^2} &= \alpha^2\frac{\partial^2 A}{\partial \mathbf{q}^2}+\alpha\beta\frac{\partial^2 B}{\partial \mathbf{q}^2}+\beta^2\frac{\partial^2 C}{\partial \mathbf{q}^2}+\alpha\frac{\partial^2 D}{\partial \mathbf{q}^2}+\beta\frac{\partial^2 E}{\partial \mathbf{q}^2}+\frac{\partial^2 F}{\partial \mathbf{q}^2},\end{aligned} \tag{14}$$

with α and β being either 0 or 1.

The computation steps outlined in **Case 1** save us three solves of Eq. (7), if we start with checking sphere-cone distances and move to sphere-sphere distance afterwards (i.e., from left to right in Fig. 4).

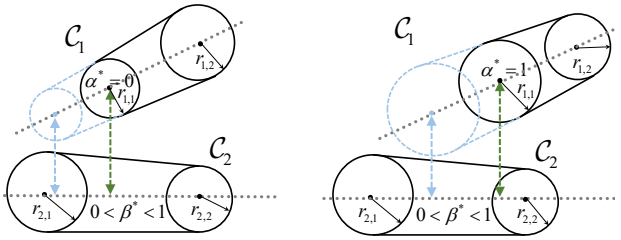

Fig. 5. **Case 2** (left) and **Case 3** (right): the global minimizer of α^* is out of the cone, which needs to be clamped, and the minimizer of β^* should be re-evaluated accordingly.

■ **Case 2**: $\alpha^*<0$, and $\beta^*\in[0,1]$.
If Eq. (10) gives $\alpha^*<0$ and $\beta^*\in[0,1]$, we have a beyond-interval minimizer (Fig. 5 left). Therefore, we need to clamp $\alpha^*=0$, which leads to $f=C\beta^2+E\beta+F$. Setting $\partial f/\partial\beta=0$ gives $\beta^*=-E/2C$. The medial distance is then computed as: $f^*=f(0,-E/2C)=F-E^2/4C$. In this case, only C, E, and F are involved in f^*, and the corresponding partial derivatives are:

$$\begin{aligned}\frac{\partial f}{\partial C}&=\frac{E^2}{4C^2}, & \frac{\partial f}{\partial E}&=-\frac{E}{2C}, & \frac{\partial f}{\partial F}&=1,\\ \frac{\partial^2 f}{\partial C^2}&=-\frac{E^2}{2C^3}, & \frac{\partial^2 f}{\partial E^2}&=-\frac{1}{2C}, & \frac{\partial^2 f}{\partial F^2}&=0.\end{aligned} \tag{15}$$

The gradient and Hessian of the medial displacement can then be obtained by substituting Eq. (15) into Eq. (13).

■ **Case 3**: $\alpha^*>1$, and $\beta^*\in[0,1]$.
As shown in Fig. 5 right, **Case 3** is similar to **Case 2**, except that the clamped sphere is at the other end of C_1. Clamping α^* to 1, β^* becomes $-(B+E)/2C$, and the medial distance is $f^*=f(1,-(B+E)/2C)=A+D+F-(B+E)^2/4C$. f^* is linear to A, D, and F. Thus, we only need to compute partial derivatives of B, C, and E as:

$$\begin{aligned}\frac{\partial f}{\partial B}&=-\frac{B+E}{2C}, & \frac{\partial f}{\partial C}&=\frac{(B+E)^2}{4C^2}, & \frac{\partial f}{\partial E}&=-\frac{B+E}{2C},\\ \frac{\partial^2 f}{\partial B^2}&=-\frac{1}{2C}, & \frac{\partial^2 f}{\partial C^2}&=-\frac{(B+E)^2}{2C^3}, & \frac{\partial^2 f}{\partial E^2}&=-\frac{1}{2C}.\end{aligned} \tag{16}$$

■ **Case 4** & **Case 5**: $\beta^*<0$ or $\beta^*>1$, and $\alpha^*\in[0,1]$.
Case 4 and **Case 5** mirror **Case 2** and **Case 3**. They can be handled in the same way as Eqs. (15) and (16), by flipping C_1 and C_2.

■ **Case 6**: $\alpha^*,\beta^*\in[0,1]$.
This case represents the most general situation as illustrated in Fig. 6. In **Case 6**, both α^* and β^* computed with Eq. (10) reside in the valid interpolation interval of $[0,1]$. Therefore, the medial distance formulation can be obtained by substituting Eq. (10) into Eq. (7):

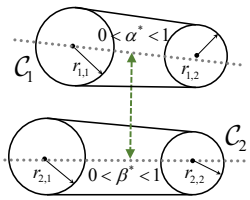

Fig. 6. **Case 6**: the medial distance is at intermediate spheres in both cones.

$$\begin{aligned}f^* &= f\left(\frac{BE-2CD}{4AC-B^2},\frac{BD-2AC}{4AC-B^2}\right)\\ &= \frac{AB^2E^2-4AC^2D^2+B^2CD^2-4A^2CE^2-B^3DE+4ABCDE}{(4AC-B^2)^2}+F,\end{aligned}$$

which can be simplified to:

$$f^*=F+\frac{BDE-(AE^2+CD^2)}{4AC-B^2}. \tag{17}$$

Its first-order partial derivatives of f^* with respect to all A to F coefficients can be computed conveniently as:

$$\begin{aligned}\frac{\partial f}{\partial A}&=\alpha^{*2}, & \frac{\partial f}{\partial B}&=\alpha^*\beta^*, & \frac{\partial f}{\partial C}&=\beta^{*2},\\ \frac{\partial f}{\partial D}&=\alpha^*, & \frac{\partial f}{\partial E}&=\beta^*, & \frac{\partial f}{\partial F}&=1.\end{aligned} \tag{18}$$

The second- and mixed-derivatives are bit involved. Here, we give the formula for coefficient A and move the rest equations for other

coefficients to Appendix B:

$$\begin{aligned}
\frac{\partial^2 f}{\partial A^2} &= -2\frac{BE-2CD}{4AC-B^2}\frac{4C(BE-2CD)}{(4AC-B^2)^2},\\
\frac{\partial^2 f}{\partial A\partial B} &= 2\frac{BE-2CD}{4AC-B^2}\left(\frac{E}{4AC-B^2}+\frac{2B(BE-2CD)}{(4AC-B^2)^2}\right),\\
\frac{\partial^2 f}{\partial A\partial C} &= -2\frac{BE-2CD}{4AC-B^2}\left(\frac{2D}{4AC-B^2}+\frac{4A}{(4AC-B^2)^2}\right),\\
\frac{\partial^2 f}{\partial A\partial D} &= -2\frac{BE-2CD}{4AC-B^2}-\frac{2C}{4AC-B^2},\\
\frac{\partial^2 f}{\partial A\partial E} &= 2\frac{BE-2CD}{4AC-B^2}\frac{B}{4AC-B^2}.
\end{aligned} \tag{19}$$

Evaluating the medial distance and its derivatives for cone-slab and slab-slab cases can be reduced to computing the medial distance for sphere-sphere (**Case 1**), sphere-cone (**Case 2** – **Case 5**), cone-cone (**Case 6**), and sphere-slab cases. When computing the sphere-slab distance, we will have one extra interpolation parameter $\gamma \in [0,1]$, corresponding to the other edge in the slab. Setting $\alpha = 0$ or $\alpha = 1$ gives us all derivatives needed, just as in Eqs. (15) and (16).

After activating necessary IPC energies and computing their gradient and Hessian for each pair of medial primitives, the simulation advances, and the solver nominates an incremental displacement of $\Delta\mathbf{q}$ at each iteration. This $\Delta\mathbf{q}$, if applied blindly, could induce inter-penetrations and fail the simulation. CCD is then invoked to avoid such risk.

5 MEDIAL CCD

In our framework, CCD is checked every time the displacement of primitives is to be updated e.g., after a Newton iteration, with a tentative $\Delta\mathbf{q}$. The primary goal of CCD is to ensure that motion trajectories of two deformable objects do not overlap, even they may appear disjointed at the end of the time step. CCD computes the first contact time a.k.a. the time of impact (TOI) between them to trim $\Delta\mathbf{q}$ so that IPC can be applied in the next iteration timely.

We keep our discussion with the example of C_1 and C_2 and add a time parameter $t \in [0,1]$ to linearize their trajectories over $\Delta\mathbf{q}$. The sphere centers $c_{i,j}(t)$ (i.e., for $i,j = \{1,2\}$) now depend on t as well. Yet, they are still interpolating all intermediate spheres in C_1 and C_2. Hence, trajectories of $c_{i,j}$ can be concisely written as:

$$c_{i,j}(t) = c_{i,j} + v_{i,j}t. \tag{20}$$

Here, $c_{i,j} = c_{i,j}(0)$ is the position of the sphere center at $t = 0$ i.e., before $\Delta\mathbf{q}$ is applied, and $v_{i,j}$ is the (constant) velocity during the normalized time interval. As the trajectory is the focus, the expression of the medial distance between C_1 and C_2 should also be modified as:

$$\min_{\alpha,\beta,t} f(t), \quad f(t) = \|c_1(\alpha,t) - c_2(\beta,t)\|^2 - (r_1(\alpha)+r_2(\beta))^2. \tag{21}$$

We re-organize Eq. (21) similarly as we did in Eq. (7) as:

$$f(\alpha,\beta,t) = A(t)\alpha^2 + B(t)\alpha\beta + C(t)\beta^2 + D(t)\alpha + E(t)\beta + F(t). \tag{22}$$

All the coefficients are now functions of t with $v_{i,j}$ involved:

$$\begin{aligned}
&V_1 = v_{1,1} - v_{1,2}, \quad V_2 = v_{2,1} - v_{2,2}, \quad V_3 = v_{1,2} - v_{2,2},\\
&C_1(t) = C_1 + V_1 t, \quad C_2(t) = C_2 + V_2 t, \quad C_3(t) = C_3 + V_3 t,\\
&A(t) = V_1^\top V_1 t^2 + 2V_1^\top C_1 t + (C_1^\top C_1 - R_1^2),\\
&B(t) = 2(V_1^\top V_2 t^2 + (V_1^\top C_2 + V_2^\top C_1)t + (C_1^\top C_2 - R_1R_2)),\\
&C(t) = V_2^\top V_2 t^2 + 2V_2^\top C_2 t + (C_2^\top C_2 - R_2^2),\\
&D(t) = 2(V_1^\top V_3 t^2 + (V_1^\top C_3 + V_3^\top C_1)t + (C_1^\top C_3 - R_1R_3)),\\
&E(t) = 2(V_2^\top V_3 t^2 + (V_2^\top C_3 + V_3^\top C_2)t + (C_2^\top C_3 - R_2R_3)),\\
&F(t) = V_3^\top V_3 t^2 + 2V_3^\top C_3 t + (C_3^\top C_3 - R_3^2).
\end{aligned} \tag{23}$$

Medial IPC employs a reduced CCD formulation checking the TOI between medial primitives based on the trajectory of medial distance defined above. Because medial distance reaches zero only if two primitives are in contact, TOI computed using medial distance is the same as the one computed using the standard Euclidean distance (i.e., Eq. (5)). TOI search for primitives is a root-finding problem of high-order polynomials, which is more expensive than triangle-based CCD processing. Fortunately, because reduced modeling enables the total number of CCD instances to be aggressively lowered, medial CCD can still remain more efficient than traditional CCD especially on GPU. For instance, in the example reported in Fig. 1, fullspace CCD needs over 700 ms with the help of spatial hashing, while medial CCD takes less than 10 ms on GPU.

5.1 Sphere-cone CCD

Sphere-cone CCD, as the name implies, checks the potential overlap between one medial sphere at the end of C_1 and C_2, or vice versa. To this end, we set α or β either 0 or 1. Mathematically, TOI in sphere-cone CCD corresponds to finding the smallest root t^*, $0 < t^* \le 1$ of four possible collision equations namely, $f(0,\beta,t) = 0$, $f(1,\beta,t) = 0$, $f(\alpha,0,t) = 0$, and $f(\alpha,1,t) = 0$.

Here, we detail the computation procedure for $f(0,\beta,t) = 0$. Other equations can be dealt with similarly. To solve $f(0,\beta,t) = 0$, we first substitute $\alpha = 0$ into Eq. (22) to have: $f(0,\beta,t) = C(t)\beta^2 + E(t)\beta + F(t) = 0$. A meaningful root $t^* \in (0,1]$ exists only when the following four conditions are satisfied simultaneously:

$$\begin{cases}
E^2(t) - 4C(t)F(t) \ge 0,\\
-E(t) \pm \sqrt{E^2(t) - 4C(t)F(t)} \ge 0,\\
-E(t) \pm \sqrt{E^2(t) - 4C(t)F(t)} \le 2C(t),\\
0 < t \le 1.
\end{cases} \tag{24}$$

Here, $E^2(t) - 4C(t)F(t) \ge 0$ ensures $f(0,\beta,t) = 0$ has real roots for β, which should also be within its interpolation interval of $[0,1]$. This requirement is enforced by $-E(t) \pm \sqrt{E^2(t) - 4C(t)F(t)} \ge 0$ and $-E(t) \pm \sqrt{E^2(t) - 4C(t)F(t)} \le 2C(t)$. Those inequations are all quartic. In our implementation, we use Ferrari's method [Herbison-Evans 1995] to retrieve all the real roots of the corresponding equations i.e., $-E(t) \pm \sqrt{E^2(t) - 4C(t)F(t)} = 0$ and $-E(t) \pm \sqrt{E^2(t) - 4C(t)F(t)} = 2C(t)$. Those roots split $(0,1]$ into multiple sub-intervals. $f(0,\beta,t)$ is either positive or negative in each sub-interval, and all the sub-intervals satisfying the inequation are collected. After that, a set intersection operation is performed over sub-intervals of all inequations in Eq. (24), which gives the final interval of $[t_{min}, t_{max}]$, if not empty. Clearly, $0 < t_{min} \le t_{max} \le 1$. The TOI is then obtained as $t^* = t_{min}$.

It is possible that $C(t) = 0$ suggesting that $f(0, \beta, t) = E(t)\beta + F(t) = 0$ is no longer a quadratic equation with respect to β. Geometrically, this occurs when the sphere at one end of the cone is fully inside the sphere at the other end. In this case, we only need to check if t^* is between 0 and 1 with $\beta = 0$ or $\beta = 1$. In other words, the sphere-cone medial distance between C_1 and C_2 further degenerates to sphere-sphere distance, which can be trivially solved. However, such situation rarely happens in practice (we never observe a single instance in our experiments).

5.2 Cone-cone CCD

The cone-cone CCD is carried out afterwards, which mimics the traditional edge-edge CCD processing on a triangle mesh, but with an escalated problem order. When two edges of C_1 and C_2 are not in parallel i.e., $4A(t)C(t) - B(t)^2 \neq 0$, we know the distance minimizers are

$$\alpha^*(t) = \frac{B(t)E(t) - 2C(t)D(t)}{4A(t)C(t) - B(t)^2}, \text{ and } \beta^*(t) = \frac{B(t)D(t) - 2A(t)E(t)}{4A(t)C(t) - B(t)^2}.$$

This is just reiterating Eq. (10), except that $\alpha^*(t)$ and $\beta^*(t)$ now represent moving trajectories over $t \in [0, 1]$. The collision condition then becomes $f(\alpha^*(t), \beta^*(t), t) = 0$, which is a hexic equation.

Similar to the sphere-cone CCD, we list two sets of constraints to keep α^* and β^* inside $(0, 1)$:

$$\begin{cases} 4A(t)C(t) - B^2(t) > 0, \\ B(t)E(t) - 2C(t)D(t) > 0, \\ B(t)D(t) - 2A(t)E(t) > 0, \\ B(t)E(t) - 2C(t)D(t) < 4A(t)C(t) - B^2(t), \\ B(t)D(t) - 2A(t)E(t) < 4A(t)C(t) - B^2(t), \\ 0 < t \leq 1, \end{cases} \tag{25}$$

and

$$\begin{cases} 4A(t)C(t) - B^2(t) < 0, \\ B(t)E(t) - 2C(t)D(t) < 0, \\ B(t)D(t) - 2A(t)E(t) < 0, \\ B(t)E(t) - 2C(t)D(t) > 4A(t)C(t) - B^2(t), \\ B(t)D(t) - 2A(t)E(t) > 4A(t)C(t) - B^2(t), \\ 0 < t \leq 1. \end{cases} \tag{26}$$

The intersection of those inequations yields another interval of (t_{min}, t_{max}). If $f(t_{min}) \leq 0$, we know a collision with an out-interval α or β occurs between $(0, t_{min}]$, which should already be handled in the sphere-cone CCD processing. Otherwise, we seek for the smallest root of $f(t^*) = 0$ for $t^* \in (t_{min}, t_{max})$. Unfortunately, there does not exist a closed-form formula for solving general hexic equations, and we resort to numerical methods to find t^*. There are many numerical root-finding algorithms such as bisection method [Ehiwario and Aghamie 2014] or ITP method [Argyros et al. 2019; Oliveira and Takahashi 2020]. As we want to find t^* just right to t_{min}, we use Householder's method with the initial guess of t_{min}. Typically, it converges within ten iterations if t^* does exist. To validate the robustness of this numerical solve, we synthesized corner cases for collision between medial primitives similar to [Erleben 2018]. Those 18 tests are reported in Fig. 7

We would like to point out that the root-finding computation is not necessary for the sphere-cone CCD. In the sphere-cone case, for any $t \in [t_{min}, t_{mas}]$, we can always find a minimizer of α^* or β^* that makes $f(t^*) = 0$, as long as constraints listed in Eq. (24) are satisfied. The global t^* is set as the smallest TOI for all primitive pairs. If $t^* \neq 1$, the step size of $\Delta\mathbf{q}$ is conservatively adjusted to $\Delta\mathbf{q} \leftarrow 0.8t^*\Delta\mathbf{q}$. Doing so guarantees the updated $\mathbf{q}$ remains in a collision-free state i.e., slightly before the global TOI. Basically, this parameter presents a trade-off between the convergence speed of current iteration and the nonlinearity of the next step. Specifically, if we set the step size very close to 1.0, two primitives will move very close to each other at the next iteration. The resulting barrier energy will also be very large yielding a large-scale repulsion force and a highly stiff system. If this parameter is set over-conservative like 0.1, $\Delta\mathbf{q}$ does not provide sufficient change to the system. Therefore, the total number of iterations increases. After $\Delta\mathbf{q}$ is committed, the medial distance needs to be updated, and IPC activations vary accordingly.

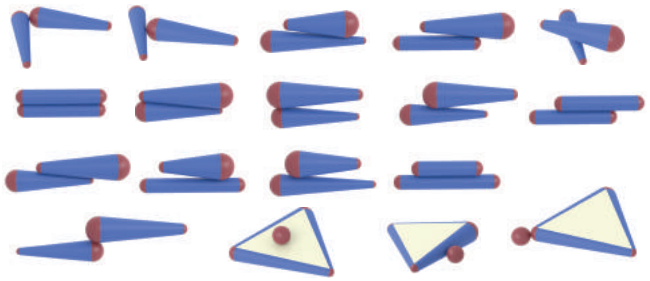

Fig. 7. Our CCD algorithm robustly returns the correct TOI in corner cases.

6 CUBATURE-FREE SUBSPACE INTEGRATION

As explained in § 4, the gradient and Hessian of reduced IPC energy can be directly expressed with the generalized coordinate. The reduced nonlinear elastic force and force gradient however, do not enjoy such convenience. This has been a major technical hurdle for reduced Lagrangian elastic simulation. A practical solution to restore the subspace efficiency is the so-called Cubature sampling [An et al. 2008]. Cubature-sampled reduced force is, however sensitive to training poses, and it casts uncertainties over the system convergence when we have many high-frequency collision-triggered deformations. For complicated models with sharp and concave local geometries, it is impractical to synthesize all the poses that will be seen in the simulation. We have little control over the error induced by such sparse force samples, and setting an appropriate convergence threshold becomes problematic, if not impossible.

Fortunately, we find that Cubature can be avoided in medial IPC. This is because the dimensionality of our subspace (in thousands) out scales other hyper-reduced models (i.e., only few dozens as in [An et al. 2008; Barbič and James 2005]). In this situation, the matrix solve with CCD-based line search stands as the dominant computation. In general, the reduced force and Hessian ($\widetilde{\mathbf{K}}$) are calculated via:

$$\widetilde{\mathbf{f}}_{int} = \mathbf{U}^\top \mathbf{f}_{int}, \quad \widetilde{\mathbf{K}} = \mathbf{U}^\top \left(\frac{\partial \mathbf{f}_{int}}{\partial \mathbf{u}} \right) \mathbf{U}. \tag{27}$$

They are standard matrix-vector and matrix-matrix products and could be significantly accelerated with cuBLAS [Nvidia 2008].

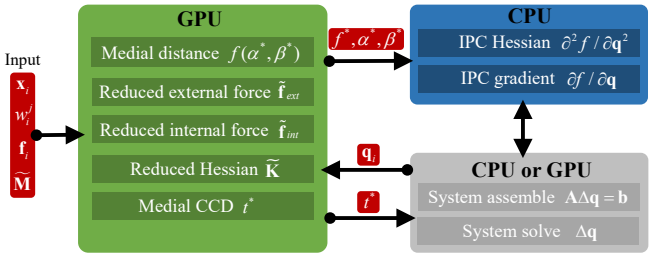

Fig. 8. An overview of our medial IPC implementation. We use CPU for sequential computations while use GPU for parallelable computations.

The real challenge is the consumption of GPU memory. Medial IPC runs with double-precision floating point arithmetics in order to accurately capture local collision events. Under such a configuration, the size of the subspace matrix could be substantial. For instance, a deformable object with 300K vertices and $n = 1,000$ will need almost 7G GPU memory for storing $\mathbf{U}$ matrix alone.

If we take a second look over the structure of medial subspace matrix $\mathbf{U}$ defined in Eq. (3), it should be immediately noticed that saving $\mathbf{U}_i$ in its native form is wasteful as the true information needed for this 3×12 matrix is only the rest position of the vertex ($\mathbf{x}_i$) and weight distributions from nearby handles. In other words, only the weight information is needed for $\mathbf{U}$ as the rest shape position of the model is already there (i.e., for rendering etc). The total memory footprint is then reduced by an order, from 13 `double` variables per handle (12 for the affine transformation and 1 for weight) to only one `double` variable.

As a pleasing by-product, such a compact $\mathbf{U}$ storage also brings a noticable performance boost for subspace integration. We note that the resulting reduced force at each vertex can be efficiently evaluated following the affine transformation form instead of using matrix-vector multiplication. In the experiment reported in Fig. 1, doing so brings over 60% speedups with `CUDA` over a "brute-force" matrix-vector or matrix-matrix based `cuBLAS` implementation. As a result, medial subspace integration only takes 20% or less of the matrix solve time, unnecessitating Cubature training. Being free of possible errors in force sampling, each Newton iteration reliably lowers the residual error. The system converges below 1% with dozens iterations even in complicated simulations.

In our implementation, the subspace Hessian integration is carried out over elements of $\widetilde{\mathbf{K}}$ in parallel. We leave the discussion with more details to the next section.

7 HETEROGENEOUS CPU-GPU IMPLEMENTATION

Unlike IPC or medial elastics that runs either solely on CPU or GPU, the implementation of medial IPC is hybrid on both computing units, aiming to fully harness the hardware power and push the simulation performance to the limit. In this section, we give a more comprehensive discussion of our implementation, and how we assign different computation tasks along the simulation pipeline to different hardware. Our overall philosophy is straightforward: moving parallelable computations to the GPU as much as possible while keeping the communication between CPU and GPU light.

7.1 Pipeline Overview

Fig. 8 sketches a high-level overview of our heterogeneous implementation. In the figure, computing tasks colored in green run on the GPU, and tasks in blue are on the CPU. Specifically, our framework takes the rest-shape vertex position ($\mathbf{x}_i$), its weights from different handles (w_i^j), and pre-built reduced mass matrix $\widetilde{\mathbf{M}}$ as input. At each time step, the simulator also receives information of external force in fullspace. We do not explicitly build a fullspace stiffness matrix $\mathbf{K} = \partial \mathbf{f}_{int} / \partial \mathbf{u} \in \mathbf{R}^{N\times N}$. Instead, 12-by-12 element stiffness matrix is assembled on the GPU in parallel. As mentioned, we adopt an altered Neo-Hookean material [Smith et al. 2018]. The PSD projection of the elastic Hessian is dealt with by clamping eigenvalues of 3-by-3 and 2-by-2 matrices, where the SVD of deformation gradient at each element is computed using the optimal SVD procedure described in [McAdams et al. 2011]. Per-element external force, elastic force, and Hessian are then projected into reduced forces and Hessian (see § 7.2).

In the meantime, with $\Delta\mathbf{q}$ computed from the previous iteration, updating the medial distance between primitives is on the GPU. At this step, model reduction brings the complexity down to the order of $\mathbf{O}(n^2)$, and the computation only takes milliseconds. Therefore, collision culling is not included in our pipeline. It is possible that, for instance when the object undergoes a large nearly-rigid motion, a high-level bounding volume could save some unneeded medial distance updates. This benefit is marginal when putting into the context of entire simulation time. After all, this paper does not aim at a real-time simulation.

After the update of medial distance f^*, together with the corresponding minimizer α^*, β^* (and γ^* for slabs), we pass this information as a $\mathbf{O}(n^2)$ array to the CPU, where the gradient and Hessian of the IPC energy is computed with multi-threading. We find this computation is better suited for CPU. The reasons are twofold. First, while the maximal number of collision events could be $\mathbf{O}(n^2)$, in reality the count of collisions among primitives is much smaller (e.g., a few hundred). The collision force preventing a pair of primitives from interpenetrating each other resolves all the potential collisions over triangles inside the primitive. In other words, the density of collision events is spatially reduced under medial IPC. This not only contributes a faster collision detection processing, more importantly it also reduces the total iterations needed for resolving the collision accurately. Secondly, we also need to perform PSD projection of the IPC hessian. This step, unfortunately cannot be simplified to eigen-problems of smaller sizes. CPU works better for this processing. We defer the discussion of PSD projection for IPC Hessian to § 7.3.

The system assembly and solve could occur at either CPU or GPU (i.e., in the form of $\mathbf{A}\Delta\mathbf{q} = \mathbf{b}$ as in Fig. 8), depending on the size of the simulation problem and the internal format of the matrix. We note that for small- and mid-scale simulation scenarios (i.e., n is in thousands), where the subspace system matrix can be directly saved as a dense matrix, GPU outperforms CPU significantly using `cuSOLVER` library. With an increased subspace size, a dense matrix representation becomes prohibitive, and the sparse matrix format should be used. In this situation, CPU takes back the lead. After the system solve, the resulting $\Delta\mathbf{q}$ is to be mollified by the medial CCD computation, which again runs on the GPU in parallel. Medial CCD

returns a global TOI t^*, and the length of $\Delta\mathbf{q}$ will be adjusted if $t^* < 1$. Next, we provide mentionable details in some key implementations along our pipeline.

7.2 Assembly of Reduced Elastic Hessian

The reduced elastic Hessian $\widetilde{\mathbf{K}}$ exhibits a block-wise structure similar to the conventional finite element stiffness matrix. Specifically, $\widetilde{\mathbf{K}}$ can be partitioned into $n \times n$ sub-matrices, and a sub-matrix is 12-by-12 encoding the elasticity coupling between corresponding handles. We first build a list of contributing elements for each 12-by-12 sub-matrix. A so-called contributing element refers to the one with at least one vertex holding a non-zero weight from the handles. The weight influence of each handle is local due to the interpolation property of biharmonics weighting. In our implementation, we discard a weight coefficient smaller than 0.001 to explicitly enhance the weight locality.

At each iteration, we first calculate updated element stiffness matrix in parallel across all the elements on the model. After that, we launch $n(n+1)/2$ blocks on `CUDA` corresponding to sub-matrices in the upper triangular part of $\widetilde{\mathbf{K}}$. Each `CUDA` block assembles the 12-by-12 sub-matrix assigned. Within the block, we also launch $12 \times (12+1)/2 = 96$ threads i.e., the upper triangular part of the sub-matrix to fully exploit the matrix symmetry. A thread computes an entry of the 12-by-12 sub-matrix by applying Alg. **??** twice. We use shared memory to minimize the latency of massive GPU memory accesses. Specifically, each block allocates two segments of shared memory. Threads first fetch and store the fullspace element stiffness matrix (which is also 12-by-12, and only 96 entries are needed) for the current element on the list. After the computation of this element finishes, all the threads move to the next element on the list. The second segment is for the reduced stiffness matrix block being built, on which the summation is performed. It is possible, for geometrically complicated models, that some handles house much more contributing elements than others residing in some sharp and salient locations on the model. Or the list just contains too many elements. In this case, to maximize the parallelization performance, we split the element list into multiple sub-lists, so that each sub-/list is of roughly the same size, and it is not too long for being processed by a single block.

We note that our implementation for reduced elastic Hessian is specially crafted for medial IPC. If the subspace size n is small, parallelizing the matrix assembly at the $n \times n$ scale may not able to fully occupy all the `CUDA` cores on the GPU. Sparse subspace matrix and localized weight distribution also greatly relieve the computation intensity of subspace integration.

7.3 PSD Projection of Distance Hessian

Similar to elastic Hessian, we need to ensure IPC Hessian is always PSD by clamping its negative singular values. If an IPC energy is between a pair of cones, its Hessian is a 48-by-48 matrix. Its dimensionality increases to 96 for a slab-slab IPC. Computing SVD for matrices of this size is expensive, and should be avoided. As one can tell from Eq. (11), the IPC energy is convex with respect to the medial distance. Therefore, PSD projection only needs to take care of the Hessian matrix of the medial distance. To better expose this fact, we re-write the distance Hessian as:

$$\frac{\partial^2 f}{\partial \mathbf{q}^2} = \frac{\partial \mathbf{c}}{\partial \mathbf{q}}^\top \left(\frac{\partial^2 f}{\partial \mathbf{c}^2} \right) \frac{\partial \mathbf{c}}{\partial \mathbf{q}}, \tag{28}$$

where we concatenate sphere centers and two ends of C_1 and C_2 into a single vector: $\mathbf{c} = [c_{1,1}^\top, c_{1,2}^\top, c_{2,1}^\top, c_{2,2}^\top]^\top \in \mathbb{R}^{12\times 1}$. Note that $\partial^2 \mathbf{c}/\partial \mathbf{q}^2$ is vanished, and $\partial \mathbf{c}/\partial \mathbf{q}$ is essentially a selection matrix, picking translational DOFs out of $\mathbf{q}$ ($\mathbf{c}$ is prescribed by the translational DOFs at relevant handles). Therefore, $\partial^2 f/\partial \mathbf{c}^2$ is the only possible source of negative definiteness, which is a 12-by-12 matrix. If the medial distance is between a sphere and a slab, $\mathbf{c}$ remains a 12-dimension vector containing the coordinate of the sphere center and three medial vertices on the slab. In our implementation, we actually compute $\partial f/\partial \mathbf{c}$ and $\partial^2 f/\partial \mathbf{c}^2$ first, instead of directly using Eq. (13). As this computation occurs on CPU, we use built-in SVD routine from `Eigen` library [Guennebaud et al. 2010] for the PSD projection. The resulting $\partial^2 f/\partial \mathbf{c}^2$ is always PSD (so the system matrix is positive definite after mass regularization) and converted to the dimension of $\mathbf{q}$ afterwards.

8 EXPERIMENTAL RESULTS

We have implemented the proposed medial IPC framework on a desktop computer with an `intel i7 9700` CPU and an `nVidia Titan RTX` GPU (with 16G GPU memory). We tested our system on a variety of geometrically complex models in collision- and contact-intense scenes. In many tests, we also compare our method with fullspace IPC simulation. We show that our system delivers significant speedups over the full simulation. The simulation results are also high-quality and indistinguishable from full IPC. As discussed in § 7, our implementation is hybrid. The system matrix is solved with `cuSOLVER` if on GPU, and `MKL PARDISO` if on CPU. We also used `Eigen` library for PSD projection and other light CPU linear algebra calculations. We refer readers to the supplementary video for more animated results.

Table 1. Geometric specifications of deformable models. # **Ele.** and # **Tri.** are total numbers of tetrahedra on the simulation mesh and triangle faces on the surface. # **MP** represents the total number of medial primitives on the reduced model, and n is the size of the subspace. The former determines the computation intensity for update medial distance and CCD, and the latter determines the time needed for reduced system solve. **MAT** gives the pre-computation time for build the initial MAT of the mode. **QM** is the computation time used for simplifying the initial MA with Q-MAT [Li et al. 2015]. **BW** is the time used to compute biharmonics weight [Jacobson et al. 2011]. All the timing records (the last three columns) are in seconds.

	# Ele.	# Tri.	# MP (n)	MAT	QM	BW
Puffer ball	625K	120K	1,324 (19,488)	776	73	13
Ship	487K	236K	610 (8,136)	640	64	4
Armadillo	95K	62K	142 (1,200)	362	6	1
Dinosaur	56K	30K	127 (1,200)	133	2	1
Cactus	693K	308K	454 (6,420)	860	60	8

8.1 Model Specifications

Tab. 1 reports specifications of 3D models in experiments. Most of deformable models we have tested are geometrically complex with concave and fine local structures. Such geometry is not friendly

for traditional modal analysis. However, medial IPC well captures their dynamic deformations because MAT intends to assign medial vertices at non-smooth locations for a better shape approximation.

To generate MAT, we need an initial Voronoi tessellation [Amenta and Bern 1999]. This step could take a few minutes. After that, we use Q-MAT [Li et al. 2015] to simplify the resulting MAT and obtain the medial mesh for handle placement. Biharmonic weights at vertices are pre-computed and are processed in parallel with multi-threading on CPU. These represent all the pre-computations of medial IPC (see the pre-computation timing information in Tab. 1), as we do not need to generate training poses and carry out Cubature training. In general, medial mesh computed automatically suffices and works well in follow-up simulations. However, one could also manually put more handles to the model to cater for specific requirements. Collision forces are explicit in medial elastics. This is not the case in medial IPC, where all the models are coupled implicitly via IPC barrier. Therefore, if a simulation scene involves multiple objects, the system size is the sum of subspace sizes of all the participating models. This echoes our original argument: model reduction is more efficacious for CCD-based simulations as you easily arrive at a simulation problem of a very high dimension.

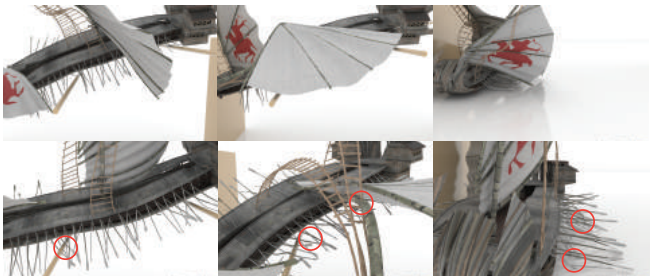

Fig. 9. When we set time step to 1/30, the simulation of the barbarian ship with medial IPC remains robust and interpenetration-free. In addition to Fig. 1, we report more simulation snapshots of medial IPC (top) and medial elastics (bottom). We can see from those zoom-in views, many collisions are not correctly resolved with DCD in medial elastics.

8.2 Collision Free with Efficiency

CCD works seamlessly with medial IPC in the reduced space. It is certainly more expensive than DCD, but in turn, CCD guarantees the simulation being collision-free at any time instance. Explicit collision handling as in medial elastics does not have this property. When we have faster-moving objects, sharp local geometries, or big time steps, artifacts like overlapping and penetration are inevitable. In the experiment shown in Fig. 1, we compare the results using medial IPC and medial elastic in the falling barbarian ship simulation to show case the difference between CCD and DCD. The ship model consists of nearly half-million elements (487K), and it falls over an array of rods. The volumetric tetrahedral mesh is created by voxelizing the input surface model [James et al. 2004]. With a conservative time step size like 1/100, the result from medial elastics is plausible and similar to ours. However, if we increase the time step size to 1/50, a zoom-in view reveals many "pass throughs" as elastic paddles at ship sides move swiftly through each other within a time step. When the mast passes the canvas, it could stay penetrated by multiple triangles on the canvas and generate ghost "shaking force" (i.e., see this artifact in the video). On the other hand, medial IPC does not have this issue. We further increase the time step to 1/30, and medial IPC consistently produce collision-free animations while more pass-throughs are observed with medial elastics (Fig. 9).

Fig. 10. The animation quality from medial IPC is close to full simulation. Readers can also find this animation in the supplementary video.

We also compared medial IPC with full IPC side by side in this experiments, and a snapshot can be found in Fig. 10. The results are very similar to each other despite that medial IPC runs two hundred times faster. We refer readers to the supplementary video for animated comparative results.

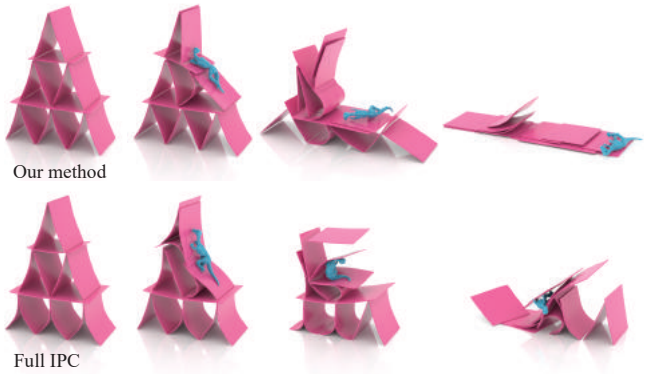

Fig. 11. Flying dinosaur into soft "house of cards". Friction can be accurately modeled with medial IPC. In this experiment, a dinosaur flies into a stack of thin elastic boards, which are stacked together under static frictions initially. Under the impact from the dinosaur, the "house of cards" crashes and falls on the floor. In this experiment, the dinosaur has 100 handles, and each board has 25 handles. Medial IPC produces highly realistic animation with 25× speedups. We also put snapshots from full IPC simulations at the bottom for the reader's reference.

8.3 Friction in Medial IPC

Following Li and colleagues [2020], we approximate friction forces defined by Maximal Dissipation Principle [Moreau 2011] by smoothing the transition between static and dynamic friction forces up to C^1-continuous, and then apply a semi-implicit temporal discretization on the tangent operator and normal force λ to define a lagged friction potential for our optimization time integrator. This provides us a robust friction model that resolves static friction with controllable accuracy as in full IPC [Li et al. 2020]. To compute the tangent

Fig. 12. The dinosaur falls into a bush of cactus. Under a large friction coefficient, the dinosaur gets stuck in the cactus (top). There are 535 handles placed at the cactus in total. If we reduce the friction coefficient, the dinosaur eventually falls on the ground (bottom). We believe this experiment show cases the an important advantage of medial IPC over PD-based medial elastics, where friction tuning is less intuitive.

operator, for a given primitive pair, we compute the medial distance and the corresponding minimizer α^*, β^*, and γ^* (for a slab). They correspond to a point on the edge of the primitive. The tangent space of the contact is then defined as the orthogonal plane at the middle point. This is consistent to the full IPC where tangent plane is directly computed from primitive relative positions. But for computing normal force λ, the barrier gradient for MAT primitives also includes ∇r i.e., the change of radius on the primitives, which could not be easily factored out. Fortunately, our distance function (Eq. 6) can be viewed as an approximation to the squared distance between primitive surfaces (Eq. 5), we can thus apply the same normal force expression of as in full IPC to compute an approximated λ by simply swapping out the distance values. Since the error to this approximation vanishes as the distance goes to 0, it does not introduce any visual inaccuracies in our experiments.

An example is reported in Fig. 11, where we have a stack of rubber boards forming a "house of cards" by static card-card friction and card-floor friction. After that, a dinosaur flies in and hit the card. This collision breaks the static friction and generates sliding frictions. We also put full simulation results in the figure for the reader's reference. Another experiment is shown in Fig. 12. The dinosaur model now jumps into a cactus bush in the desert (total 535 handles). With high friction coefficient, the dinosaur model is stuck in the cactus while the cactus branches are interacting each other intensively. If we reduce the friction coefficient, we can see the dinosaur eventually falls on the ground. Such friction-involving effect is not native to PD-based method like medial elastics.

8.4 How Much We Should Reduce

As a reduced simulation method, medial IPC seeks for the converging spot between simulation quality and computation performance, with the presence of CCD and implicit collision resolve. To this end, we would like to investigate, at least visually, the relation between

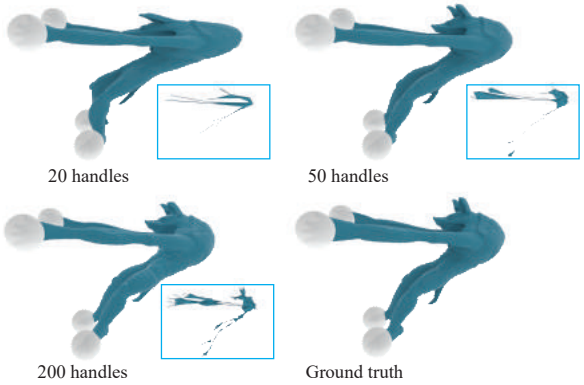

Fig. 13. We compare how does the sparsity of the handle placement influence the final deformation effect. An over-aggressive reduction leads to artifacts. For instance, when we only have 20 handles without nonlinear simulation DOFs, it can be seen that medial IPC tries to use large local shearing to "mimic" the bending at the back. Fortunately, this artifact is greatly relieved with more handles. The simulation under 50 or 200 handles is very close to the ground truth, where the system solve only needs dozens of milliseconds.

the handle sparsity (subspace size) and final animation quality. Another motivation for this experiment is the lack of nonlinear DOFs in medal IPC. Unlike some existing work [Lan et al. 2020; Luo et al. 2018; Martin et al. 2010], we do not put quadratic handles on the medial mesh due to the concern of GPU memory footprint (i.e., as discussed in § 6). As a counterstrategy in medial IPC, we try to add more linear handles to provide sufficient freedom to the animation. In Fig. 13, we report an experiment, where we fix hands and feet of the Armadillo and apply a large force at its back to trigger a sharp deformation. We compare the deformed poses simulated with 20,

50, and 200 handles. We can clearly observe artifacts when the total number of handle is limited (e.g., 20). Populating more handles is quite effective in improving the visual effects. When we have a bit more handles e.g., 50 handles added, the results look very similar to the ground truth. In our other experiments, we normally have much more handles placed on the model.

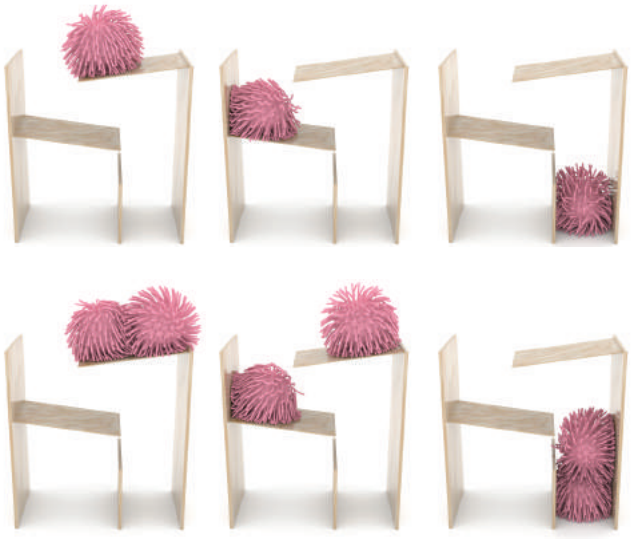

Fig. 14. In this stress test, we drop a puffer ball into a zigzag wooden shelf. There are 625K elements and 1,624 handles on the puffer ball. The subspace size of one puffer ball is around 20K. In this experiment, `i7 9700` outperforms `Titan` GPU by 75% (1.2 vs. 2.1 seconds). A similar performance lead takes place for CPU after we add one more puffer ball into the simulation (4.3 vs. 2.8 seconds). Regardless, medial IPC is 36× and 26× faster than the full simulation in those two settings respectively.

8.5 CPU/GPU Performance vs. System Size

As discussed in Fig. 8, medial IPC runs on both GPU and CPU in an coordinated way. The medial distance update, element-wise matrix assembly and subspace integration are all carried out on GPU. Distance gradient/Hessian and its PSD projection are on CPU. The undecided component, and also the most expensive computation, is the system solve, which is either on CPU or GPU. Because our system matrix is always positive definite with PSD Hessian, we use Cholesky factorization shipped with `cuSOLVER` or `MKL` to solve the system matrix on GPU or CPU. The question is which platform should we choose?

In order to better explain this question, we design a challenging simulation scene: dropping a puffer ball with hundreds of elastic strings on its surface to a zigzag wooden shelf. The result of the simulation is reported in Fig. 14 (top). The puffer ball has 610K elements with 1,624 handles placed: each string has five handles, and we put 24 handles on its main spheroid body. The reduced system size of a single such puffer ball is close to 20K ($1,624 \times 12 = 19,488$). That is bigger than two barbarian ships. We not only compare the animation effects with full FEM/IPC simulation, but also record the system solve time for this experiment on CPU and GPU. We find that `Titan` GPU gets outperformed by `i7 9700` at this problem scale, where it takes 2.1 seconds on the GPU, and only 1.2 seconds on the CPU.

Next, we include two puffer balls into the scene (Fig. 14 (bottom)). The added puffer ball significantly enriches the deformation effects. Because of implicit IPC, the system to be solved at each Newton iteration is also doubled at around 40K (38,976). Again, `MKL` on `i7 9700` holds a big lead for the system solve (4.3 seconds on the GPU vs. 2.8 seconds on the CPU). In Fig. 15, we further push the problem size to include five puffer balls. In this simulation, subspace size reaches 10K, which is bigger than many full simulation problems. The lead of CPU has been narrowed, and the system solve time is 6.9 seconds on GPU and 6.5 seconds on CPU.

Fig. 15. When we push medial IPC to simulate five puffer balls, the subspace size is almost 100K. In this case, we choose CPU for system solve. In this extreme reduced simulation, medial IPC is 14× faster.

We may conclude from those observations that GPU is best at carrying out computations that are straightforwardly parallelizable, such as matrix multiplication subspace projection, per-element matrix assembly etc. For general and sequential computations e.g., matrix factorization, the speedup of the GPU is not "native", and it over-consumes the hardware resource (massive need for temporary memory, shared memory, register assignment, etc.), which is only possible for problems of moderate sizes. This makes GPU a perfect platform for reduced simulation. After all, subspace space of 10K is rare in practice, and we observe excellent performance boost for smaller subspaces of regular sizes.

In this set of experiments, another noteworthy implementation detail is the storage of the subspace system matrix. In general, a reduced model uses dense matrix to represent reduced elastic Hessian and system matrix. We follow this principle in medial IPC as well. However, when we have multiple models in the simulation, the total size of the system is multiplied, and dense system matrix becomes prohibitive. To this end, we save the system matrix with CSR form explicitly as a sparse matrix whenever we have multiple objects in the simulation. This may be another reason behind slower performance on GPU. In the experiment shown in Fig. 14 bottom, `Titan` fails to process the resulting system matrix if we just use the dense format.

8.6 Time Profiling

We now give detailed timing information of the medial IPC pipeline for experiments in Figs 1, 11, 14, and 15 in Tab. 2. Medial IPC uses the medial primitive as the collision processing unit, which becomes much more efficient on GPU. Because CCD must be used at each

Table 2. We report time statistics for each major step along the medial IPC pipeline. We give both CPU and GPU performance. **# Barrier** gives the total number of primitive pair CCD performed, and the average number of colliding pairs. In medial IPC, primitives are never in physical contact. The number is actually the count of active barrier energies between primitives. **Ele. stiff.** is the total time used for assembling element stiffness matrix (in milliseconds). **Reduced int.** reports the total time used for subspace force and Hessian integration (in seconds). ∇^2 **IPC** (in milliseconds) is the time used for computing IPC Hessian and gradient. This computation is on CPU. **Medial dist.** is the time used for updating medial distance between primitives (in milliseconds). **CCD** (in milliseconds) is CCD processing time. **Solve/it.** is the time for solving the linearized system. It could be either on GPU or CPU. **# it.** is the average iteration numbers needed in the example. **Time/frame** (in seconds) is the total time on average for one frame of the simulation.

	# Barrier	Ele. stiff.	Reduced int.	∇^2 IPC	Medial dist.	CCD	Solve/it.	# it.	Time/frame
Fig. 1	1.4M | 188	36.4 | 4.4K	2.4 | 46.8	6.5	3.3 | 182.1	23.6 | 5.7K	1.8	15	10.1 (110×)
Fig. 11	0.32M | 536	11.2 | 0.2K	0.1 | 9.8	1.7	1.2 | 34.6	14.5 | 1.1K	0.12	10	2.6 (25×)
Fig. 12	0.28M | 89	28.4 | 3.0K	1.7 | 32.7	3.6	< 1 | 39.8	7.7 | 1.2K	0.54	9	20.8 (38×)
Fig. 14 (top)	0.95M | 1.9K	31.5 | 2.8K	1.1 | 13.1	9.3	2.1 | 115.2	40.6 | 3.8K	1.2	8	19.4 (36×)
Fig. 14 (bottom)	3.9M | 3.5K	54.3 | 5.5K	1.9 | 26	25.6	4.7 | 463.6	161.2 | 15.8K	2.8	13	65.2 (26×)
Fig. 15	24.3M | 11.7K	147.6 | 14.4K	4.1 | 67.2	131.5	7.8 | 3.0K	353.4 | 102.9K	6.6	18	207.3 (14×)

Newton iteration, this acceleration is scaled by total iteration needed. In the table, we report both CPU and GPU running time as well as the fullspace solve time for readers' reference.

The acceleration mechanism in medial IPC is similar to other model reduction frameworks, where the acceleration rate is dominantly reflected by the fullspace-subspace ratio. In the barbarian ship shown in Fig. 1, this ratio is over 50 : 1, which is the highest in all our experiments, and we receive a one-hundred-time speedup. Meanwhile, the ratio for one puffer ball simulation is only at 22 : 1, and medial IPC acceleration is weakened too. In addition, when the system matrix must be explicitly saved as a sparse matrix and solved at CPU, the acceleration rate further drops (e.g., in Fig. 15).

9 CONCLUSION AND LIMITATION

In this paper, we present a reduced simulation framework, and we name it as medial IPC. From a high level, medial IPC integrates most desired features of medial elastics and IPC into unified framework. Specifically, our subspace is built based on the MAT, which is able to capture rich and interesting deformable effects. We do not follow a projective dynamics but stick with classic Lagrangian mechanics. Therefore, medial IPC not only models real-world hyperelastic materials but also robustly deals with implicit contact in complicated simulation problems. In the meantime, we follow the concept of barrier function, combined with CCD in line search, in the IPC to process collision and contact implicitly. While this overall idea is straightforward, as we have shown we address several tough technical challenges in this system. All the computations are directly based on the reduced coordinate, which avoids unnecessary fullspace-subspace projections. We propose a novel GPU implementation that removes the Cubature training as in other model reduction systems. Medial IPC is carefully engineered with a hybrid GPU-GPU implementation. We have shown compelling animation results from our method, which is similar to the one obtained from a full simulation but the simulation time is shortened by orders.

Our system also has some limitations. First of all, our system is carefully crafted for MAT and IPC. This is a double-edge blade: at one hand, we fully optimize our implementation with dedicated CPU and GPU coordination; on the other hand, such implementation is less general and may not be applicable for other model reduction frameworks. For instance, our subspace matrix has a unique sparse structure (e.g., see Eq. (3)), if the subspace matrix is constructed by other methods like modal analysis [Pentland and Williams 1989] or component mode synthesis [Yang et al. 2015], the subspace force and Hessian integration could be much slower, and the GPU acceleration should be deployed in a different way. Medial IPC idealizes the collision model in two ways. 1) Medial IPC ignores within-primitive self-collision. We consider a primitive a basic *deformation unit* and assume triangles within a primitive do not collide. While such assumption is reasonable in most situations, it also shares an inherent limitation of medial elastics. The inconsistence of medial primitives and deformed vertices positions (the former is linearly interpolated and the latter is blended using harmonics weights) may fail the collision detection under extreme deformations. 2) Medial IPC also assumes that the radii of primitives are unchanged. With hundreds of handles, MAT is able to tightly encapsulate the deformed shape without the radius update. However, if one really needs to push the accuracy of between-primitive collisions, it is also possible to evaluate the derivative of medial distance with respect to the radius. Following the idea of barrier function, medial IPC is as robust as full IPC in the simulation. On the downside, it also fails the solver if a penetration occurs, which may be an outside error from the simulation input. We do not have any mechanisms to fix the barrier function under a negative medial distance. Medial IPC is not profitable under any subspace sizes. However, we believe this is a common drawback for model reduction methods – no one works perfectly by default for any situations.

ACKNOWLEDGMENTS

We thank anonymous reviewers for their detailed and constructive comments. Yin Yang is partially supported by National Science Foundation (NSF) under grants No. 2031002, 2011471 and 2016414, as well as Air Force Research Laboratory (AFRL) under agreement number FA9453-18-0022. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Chenfanfu Jiang and Minchen Li are partially supported by NSF under grants No. 1943199, 1813624, and 2023780. Junfeng Yao is partially supported by National Nature Science Foundation of China (No. 62072388), the industry guidance

project foundation of science technology bureau of Fujian province (No. 2019C0021, No. 2020H0047), and natural science foundation of science technology bureau of Fujian province (No. 2019J01601).

REFERENCES

Samantha Ainsley, Etienne Vouga, Eitan Grinspun, and Rasmus Tamstorf. 2012. Speculative parallel asynchronous contact mechanics. *ACM Trans. Graph. (TOG)* 31, 6 (2012), 1–8.

Nina Amenta and Marshall Bern. 1999. Surface reconstruction by Voronoi filtering. *Discrete & Computational Geometry* 22, 4 (1999), 481–504.

Steven S An, Theodore Kim, and Doug L James. 2008. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph. (TOG)* 27, 5 (2008), 1–10.

IK Argyros, MA Hernández-Verón, and MJ Rubio. 2019. On the Convergence of Secant-Like Methods. In *Current Trends in Mathematical Analysis and Its Interdisciplinary Applications*. Springer, 141–183.

David Baraff. 1994. Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of the 21st annual conference on Computer graphics and interactive technique*. ACM, 23–34.

Jernej Barbič and Doug L James. 2010. Subspace self-collision culling. *ACM Trans. Graph. (TOG)* 29, 4 (2010), 81.

Jernej Barbič and Jovan Popović. 2008. Real-time control of physically based simulations using gentle forces. *ACM Trans. Graph. (TOG)* 27, 5 (2008), 1–10.

Jernej Barbič and Doug L James. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Trans. Graph. (TOG)* 24, 3 (2005), 982–990.

Adam W Bargteil and Elaine Cohen. 2014. Animation of deformable bodies with quadratic bézier finite elements. *ACM Trans. Graph. (TOG)* 33, 3 (2014), 27.

Gino van den Bergen. 1997. Efficient collision detection of complex deformable models using AABB trees. *Journal of graphics tools* 2, 4 (1997), 1–13.

Harry Blum. 1967. A transformation for extracting new descriptors of shape. *Models for Perception of Speech and Visual Forms, 1967* (1967), 362–380.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective dynamics: fusing constraint projections for fast simulation. *ACM Trans. Graph. (TOG)* 33, 4 (2014), 154:1–154:11.

Christopher Brandt, Elmar Eisemann, and Klaus Hildebrandt. 2018. Hyper-reduced projective dynamics. *ACM Trans. Graph. (TOG)* 37, 4 (2018), 1–13.

Tyson Brochu, Essex Edwards, and Robert Bridson. 2012. Efficient geometrically exact continuous collision detection. *ACM Trans. on Graph. (TOG)* 31, 4 (2012), 1–7.

Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. 2002. Interactive skeleton-driven dynamic deformations. *ACM Trans. Graph. (TOG)* 21, 3 (2002), 586–593.

Min Gyu Choi and Hyeong-Seok Ko. 2005. Modal warping: Real-time simulation of large rotational deformation and manipulation. *IEEE Trans. on Visualization and Computer Graphics* 11, 1 (2005), 91–101.

Dirk Den Hertog. 2012. *Interior point approach to linear, quadratic and convex programming: algorithms and complexity*. Vol. 277. Springer Science & Business Media.

JC Ehiwario and SO Aghamie. 2014. Comparative study of bisection, Newton-Raphson and secant methods of root-finding problems. *IOSR J. of Engineering* 4, 4 (2014), 1–7.

Kenny Erleben. 2018. Methodology for assessing mesh-based contact point methods. *ACM Trans. Graph. (TOG)* 37, 3 (2018), 1–30.

François Faure, Benjamin Gilles, Guillaume Bousquet, and Dinesh K Pai. 2011. Sparse meshless models of complex deformable solids. *ACM Trans. Graph. (TOG)* 30, 4 (2011), 73.

Marco Fratarcangeli, Valentina Tibaldo, and Fabio Pellacini. 2016. Vivace: A practical gauss-seidel method for stable soft body dynamics. *ACM Trans. Graph. (TOG)* 35, 6 (2016), 1–9.

Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. 2020. ADD: analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Trans. Graph. (TOG)* 39, 6 (2020), 1–15.

Benjamin Gilles, Guillaume Bousquet, Francois Faure, and Dinesh K Pai. 2011. Frame-based elastic models. *ACM Trans. Graph. (TOG)* 30, 2 (2011), 15.

Stefan Gottschalk, Ming C Lin, and Dinesh Manocha. 1996. OBBTree: A hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 171–180.

Gaël Guennebaud, Benoit Jacob, et al. 2010. Eigen. *URl: http://eigen. tuxfamily. org* (2010).

David Harmon, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. 2008. Robust treatment of simultaneous collisions. *ACM Trans. Graph. (TOG)* 27, 3 (2008), 1–4.

Kris K Hauser, Chen Shen, and James F O'Brien. 2003. Interactive deformation using modal analysis with constraints.. In *Graphics Interface*, Vol. 3. 16–17.

Florian Hecht, Yeon Jin Lee, Jonathan R Shewchuk, and James F O'Brien. 2012. Updated sparse cholesky factors for corotational elastodynamics. *ACM Trans. Graph. (TOG)* 31, 5 (2012), 123.

Don Herbison-Evans. 1995. Solving quartics and cubics for graphics. In *Graphics Gems V*. Elsevier, 3–15.

Philip Martyn Hubbard. 1995. Collision detection for interactive graphics applications. *IEEE Trans. on Visualization and Computer Graphics* 1, 3 (1995), 218–230.

Geoffrey Irving, Joseph Teran, and Ronald Fedkiw. 2004. Invertible finite elements for robust simulation of large deformation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 131–140.

Alec Jacobson, Ilya Baran, Jovan Popovic, and Olga Sorkine. 2011. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph. (TOG)* 30, 4 (2011), 78–1.

Doug L James, Jernej Barbič, and Christopher D Twigg. 2004. Squashing cubes: Automating deformable model construction for graphics. In *ACM SIGGRAPH 2004 Sketches*. 38.

Doug L James and Dinesh K Pai. 2004. BD-tree: output-sensitive collision detection for reduced deformable models. *ACM Trans. Graph. (TOG)* 23, 3 (2004), 393–398.

Theodore Kim and David Eberle. 2020. Dynamic deformables: implementation and production practicalities. In *ACM SIGGRAPH 2020 Courses*. 1–182.

Theodore Kim and Doug L James. 2009. Skipping steps in deformable simulation with online model reduction. *ACM Trans. Graph. (TOG)* 28, 5 (2009), 123.

Shankar Krishnan, M Gopi, M Lin, Dinesh Manocha, and A Pattekar. 1998. Rapid and accurate contact determination between spline models using ShellTrees. *Computer Graphics Forum* 17, 3 (1998), 315–326.

Lei Lan, Ran Luo, Marco Fratarcangeli, Weiwei Xu, Huamin Wang, Xiaohu Guo, Junfeng Yao, and Yin Yang. 2020. Medial elastics: efficient and collision-ready deformation via medial axis transform. *ACM Trans. Graph. (TOG)* 39, 3 (2020), 1–17.

Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M Kaufman. 2020. Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. *ACM Trans. Graph. (TOG)* 39, 4 (2020).

Pan Li, Bin Wang, Feng Sun, Xiaohu Guo, Caiming Zhang, and Wenping Wang. 2015. Q-Mat: Computing medial axis transform by quadratic error minimization. *ACM Trans. Graph. (TOG)* 35, 1 (2015), 8.

Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. 2013. Fast simulation of mass-spring systems. *ACM Trans. Graph. (TOG)* 32, 6 (2013), 214:1–214:7.

Ran Luo, Weiwei Xu, Huamin Wang, Kun Zhou, and Yin Yang. 2018. Physics-based quadratic deformation using elastic weighting. *IEEE Trans. on Visualization and Computer Graphics* 24, 12 (2018), 3188–3199.

Miles Macklin, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Tae-Yong Kim. 2020. Primal/dual descent methods for dynamics. *Computer Graphics Forum* 39, 8 (2020), 89–100.

Miles Macklin, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Viktor Makoviychuk. 2019. Non-smooth newton methods for deformable multi-body dynamics. *ACM Trans. Graph. (TOG)* 38, 5 (2019), 1–20.

Sebastian Martin, Peter Kaufmann, Mario Botsch, Eitan Grinspun, and Markus Gross. 2010. Unified simulation of elastic rods, shells, and solids. *ACM Trans. Graph. (TOG)* 29, 4 (2010), 39.

Aleka McAdams, Andrew Selle, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. 2011. *Computing the singular value decomposition of 3x3 matrices with minimal branching and elementary floating point operations*. Technical Report. University of Wisconsin-Madison Department of Computer Sciences.

Matthew Moore and Jane Wilhelms. 1988. Collision detection and response for computer animation. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, Vol. 22. ACM, 289–298.

Jean Jacques Moreau. 2011. On unilateral constraints, friction and plasticity. In *New variational techniques in mathematical physics*. Springer, 171–322.

Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. 2005. Meshless deformations based on shape matching. *ACM Trans. Graph. (TOG)* 24, 3 (2005), 471–478.

CUDA Nvidia. 2008. Cublas library. *NVIDIA Corporation, Santa Clara, California* 15, 27 (2008), 31.

IFD Oliveira and RHC Takahashi. 2020. An enhancement of the bisection method average performance preserving minmax optimality. *ACM Transactions on Mathematical Software (TOMS)* 47, 1 (2020), 1–24.

Miguel A Otaduy, Rasmus Tamstorf, Denis Steinemann, and Markus Gross. 2009. Implicit contact handling for deformable objects. *Computer Graphics Forum* 28, 2 (2009), 559–568.

Alex Pentland and John Williams. 1989. Good vibrations: Modal dynamics for graphics and animation. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, Vol. 23. ACM.

Eftychios Sifakis and Jernej Barbic. 2012. FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses*. ACM, 20.

Breannan Smith, Fernando De Goes, and Theodore Kim. 2018. Stable neo-hookean flesh simulation. *ACM Trans. Graph. (TOG)* 37, 2 (2018), 1–15.

Feng Sun, Yi-King Choi, Yizhou Yu, and Wenping Wang. 2016. Medial meshes: a compact and accurate representation of medial axis transform. *IEEE Trans. on Visualization and Computer Graphics* 22, 3 (2016), 1278–1290.

Rasmus Tamstorf, Toby Jones, and Stephen F McCormick. 2015. Smoothed aggregation multigrid for cloth simulation. *ACM Trans. Graph. (TOG)* 34, 6 (2015), 245.
Huamin Wang and Yin Yang. 2016. Descent methods for elastic body simulation on the GPU. *ACM Trans. Graph. (TOG)* 35, 6 (2016), 212.
Hongyi Xu, Funshing Sin, Yufeng Zhu, and Jernej Barbič. 2015. Nonlinear material design using principal stretches. *ACM Trans. Graph. (TOG)* 34, 4 (2015), 1–11.
Yin Yang, Dingzeyu Li, Weiwei Xu, Yuan Tian, and Changxi Zheng. 2015. Expediting precomputation for reduced deformable simulation. *ACM Trans. Graph. (TOG)* 34, 6 (2015).
Gabriel Zachmann. 2002. Minimal hierarchical collision detection. In *ACM symposium on Virtual reality software and technology*. ACM, 121–128.
Gabriel Zachmann and Elmar Langetepe. 2003. *Geometric data structures for computer graphics*. Eurographics Assoc.
Changxi Zheng and Doug L James. 2012. Energy-based self-collision culling for arbitrary mesh deformations. *ACM Trans. Graph. (TOG)* 31, 4 (2012), 98.
Yongning Zhu, Eftychios Sifakis, Joseph Teran, and Achi Brandt. 2010. An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Trans. Graph. (TOG)* 29, 2 (2010), 16.
Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, Olgierd Cecil Zienkiewicz, and Robert Lee Taylor. 1977. *The finite element method*. Vol. 36. McGraw-hill London.

A $\partial X/\partial \mathbf{q}$ AND $\partial^2 X/\partial \mathbf{q}^2$

In this appendix, we give formulations of first- and second-order partial derivatives of X (i.e., A to F coefficients of Eq. (7) in § 4). It is not difficult to see from Eqs. (8) and (9) that those coefficients are quadratic functions of sphere centers on C_1 and C_2, and the latter are prescribed by the generalized coordinate $\mathbf{q}$. Hence, all the coefficients have linear gradients and constant Hessians.

Let $\mathbf{q} = \left[\mathbf{q}^{1,1}, \mathbf{q}^{1,2}, \mathbf{q}^{2,1}, \mathbf{q}^{2,2}\right]^\top \in \mathbb{R}^{48}$ be the generalized coordinate of four handles at C_1 and C_2. The derivatives of X with respect to other handle DOFs are clearly zero. $\mathbf{C}_i = \left[C_i^\top, \mathbf{0}_{1\times 9}\right]$, for $i = \{1, 2, 3\}$ is a 12-dimension *row* vector with last nine elements zero padded, and C_i is given in Eq. (8). $\mathbf{Q} = [2\mathbf{I}, \mathbf{0}_{3\times 9}]$. The first-order derivatives of $\partial X/\partial \mathbf{q}$ are:

$$\frac{\partial A}{\partial \mathbf{q}} = [2\mathbf{C}_1, -2\mathbf{C}_1, 0, 0], \quad \frac{\partial B}{\partial \mathbf{q}} = [2\mathbf{C}_2, -2\mathbf{C}_2, -2\mathbf{C}_1, 2\mathbf{C}_1],$$
$$\frac{\partial C}{\partial \mathbf{q}} = [0, 0, -2\mathbf{C}_2, 2\mathbf{C}_2], \quad \frac{\partial D}{\partial \mathbf{q}} = [2\mathbf{C}_3, 2(\mathbf{C}_1 - \mathbf{C}_3), 0, -2\mathbf{C}_1],$$
$$\frac{\partial E}{\partial \mathbf{q}} = [0, 2\mathbf{C}_2, -2\mathbf{C}_3, -2(\mathbf{C}_2 - \mathbf{C}_3)], \quad \frac{\partial F}{\partial \mathbf{q}} = [0, 2\mathbf{C}_3, 0, -2\mathbf{C}_3].$$

The Hessian matrices can then be obtained by:

$$\frac{\partial^2 A}{\partial \mathbf{q}^2} = \begin{bmatrix} \mathbf{Q} & -\mathbf{Q} & 0 & 0 \\ -\mathbf{Q} & \mathbf{Q} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \frac{\partial^2 B}{\partial \mathbf{q}^2} = \begin{bmatrix} 0 & 0 & -\mathbf{Q} & \mathbf{Q} \\ 0 & 0 & \mathbf{Q} & -\mathbf{Q} \\ -\mathbf{Q} & \mathbf{Q} & 0 & 0 \\ \mathbf{Q} & -\mathbf{Q} & 0 & 0 \end{bmatrix},$$

$$\frac{\partial^2 C}{\partial \mathbf{q}^2} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{Q} & -\mathbf{Q} \\ 0 & 0 & -\mathbf{Q} & \mathbf{Q} \end{bmatrix}, \quad \frac{\partial^2 D}{\partial \mathbf{q}^2} = \begin{bmatrix} 0 & \mathbf{Q} & 0 & -\mathbf{Q} \\ \mathbf{Q} & -\mathbf{Q} & 0 & \mathbf{Q} \\ 0 & 0 & 0 & 0 \\ -\mathbf{Q} & \mathbf{Q} & 0 & 0 \end{bmatrix},$$

$$\frac{\partial^2 E}{\partial \mathbf{q}^2} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -\mathbf{Q} & \mathbf{Q} \\ 0 & -\mathbf{Q} & 0 & \mathbf{Q} \\ 0 & \mathbf{Q} & \mathbf{Q} & -\mathbf{Q} \end{bmatrix}, \quad \frac{\partial^2 F}{\partial \mathbf{q}^2} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \mathbf{Q} & 0 & -\mathbf{Q} \\ 0 & 0 & 0 & 0 \\ 0 & -\mathbf{Q} & 0 & \mathbf{Q} \end{bmatrix}.$$

B $\partial^2 f/\partial X^2$ FOR **CASE 6** IN § 4.2

We give all the second and mixed derivatives of the cone-cone medial distance with respect to coefficients B to E (i.e., in Eq. (7)).

B.1 Second-derivatives of coefficient B

$$\frac{\partial^2 f}{\partial B \partial A} = \frac{BD - 2AE}{4AC - B^2}\left(-\frac{4C(BE - 2CD)}{(4AC - B^2)^2}\right) + \frac{BE - 2CD}{4AC - B^2}\left(-\frac{2E}{4AC - B^2} - \frac{4C}{(4AC - B^2)^2}\right),$$
$$\frac{\partial^2 f}{\partial B^2} = \frac{BD - 2AE}{4AC - B^2}\left(\frac{E}{4AC - B^2} + \frac{2B(BE - 2CD)}{(4AC - B^2)^2}\right) + \frac{BE - 2CD}{4AC - B^2}\left(\frac{D}{4AC - B^2} + \frac{2B(BD - 2AE)}{(4AC - B^2)^2}\right),$$
$$\frac{\partial^2 f}{\partial B \partial C} = \frac{BD - 2AE}{4AC - B^2}\left(-\frac{2D}{4AC - B^2} - \frac{4A}{(4AC - B^2)^2}\right) + \frac{BE - 2CD}{4AC - B^2}\left(-\frac{4A(BD - 2AE)}{(4AC - B^2)^2}\right),$$
$$\frac{\partial^2 f}{\partial B \partial D} = \frac{BD - 2AE}{4AC - B^2}\left(-\frac{2C}{4AC - B^2}\right) + \frac{BE - 2CD}{4AC - B^2}\left(\frac{B}{4AC - B^2}\right),$$
$$\frac{\partial^2 f}{\partial B \partial E} = \frac{BD - 2AE}{4AC - B^2}\left(\frac{B}{4AC - B^2}\right) + \frac{BE - 2CD}{4AC - B^2}\left(-\frac{2A}{4AC - B^2}\right).$$

B.2 Second-derivatives of coefficient C

$$\frac{\partial^2 f}{\partial C \partial A} = 2\frac{BD - 2AE}{4AC - B^2}\left(-\frac{2E}{4AC - B^2} - \frac{4C}{(4AC - B^2)^2}\right),$$
$$\frac{\partial^2 f}{\partial C \partial B} = 2\frac{BD - 2AE}{4AC - B^2}\left(\frac{D}{4AC - B^2} + \frac{2B(BD - 2AE)}{(4AC - B^2)^2}\right),$$
$$\frac{\partial^2 f}{\partial C^2} = 2\frac{BD - 2AE}{4AC - B^2}\left(-\frac{4A(BD - 2AE)}{(4AC - B^2)^2}\right),$$
$$\frac{\partial^2 f}{\partial C \partial D} = 2\frac{BD - 2AE}{4AC - B^2}\left(\frac{B}{4AC - B^2}\right),$$
$$\frac{\partial^2 f}{\partial C \partial E} = 2\frac{BD - 2AE}{4AC - B^2}\left(-\frac{2A}{4AC - B^2}\right).$$

B.3 Second-derivatives of coefficient D

$$\frac{\partial^2 f}{\partial D \partial A} = -\frac{4C(BE - 2CD)}{(4AC - B^2)^2},$$
$$\frac{\partial^2 f}{\partial D \partial B} = \frac{E}{4AC - B^2} + \frac{2B(BE - 2CD)}{(4AC - B^2)^2},$$
$$\frac{\partial^2 f}{\partial D \partial C} = -\frac{2D}{4AC - B^2} - \frac{4A}{(4AC - B^2)^2},$$
$$\frac{\partial^2 f}{\partial D^2} = -\frac{2C}{4AC - B^2}, \quad \frac{\partial^2 f}{\partial D \partial E} = \frac{B}{4AC - B^2}.$$

B.4 Second-derivatives of coefficient E

$$\frac{\partial^2 f}{\partial E \partial A} = -\frac{2E}{4AC - B^2} - \frac{4C}{(4AC - B^2)^2},$$
$$\frac{\partial^2 f}{\partial E \partial B} = \frac{D}{4AC - B^2} + \frac{2B(BD - 2AE)}{(4AC - B^2)^2},$$
$$\frac{\partial^2 f}{\partial E \partial C} = -\frac{4A(BD - 2AE)}{(4AC - B^2)^2},$$
$$\frac{\partial^2 f}{\partial E \partial D} = \frac{B}{4AC - B^2}, \quad \frac{\partial^2 f}{\partial E^2} = -\frac{2A}{4AC - B^2}.$$