Medial Elastics: Efficient and Collision-ready Deformation via Medial Axis Transform

ANONYMOUS AUTHOR(S)

Fig. 1. Our framework couples spatial model reduction and collision detection via medial axis transform, a high-fidelity and compact volumetric shape representation. This representation (known as the topological skeleton) does not only provide us an expressive subspace, but also serves as a tight volume enclosure to facilitate the collision detection and handling. In this figure, the barbarian ship has $482k$ elements and $236k$ triangles. Following a semi-reduced projective dynamics formulation, this model is reduced to a 2, 526-dimension subspace with $482k$ constraints. The animation runs at $\sim$ 15 FPS with all the collisions and self-collisions resolved and rich local details preserved.

We propose a framework for the interactive simulation of nonlinear deformable objects. The primary feature of our system is the seamless integration of deformable simulation and collision culling, which are often independently handled in existing animation systems. The bridge connecting them is the medial axis transform or MAT, a high-fidelity volumetric approximation of complex 3D shapes. From the physics simulation perspective, MAT leads to an expressive and compact reduced nonlinear model. We employ a semi-reduced projective dynamics formulation, which well captures high-frequency local deformations of high-resolution models while retaining a low computation cost. Our key observation is that the most compelling (nonlinear) deformable effects are enabled by the local constraints projection, which should not be aggressively reduced. The global stage solves a linear system and is less GPU-friendly, to which the model reduction applies with marginal compromise of the visual plausibility. From the collision detection/culling perspective, MAT is geometrically versatile using linear-interpolated spheres (i.e. the so-called medial primitives) to approximate the boundary of the input model. For instance, only 87 medial primitives are able to encapsulate the dinosaur model as tightly as one using 53, 294 AABBs or 37, 282 bounding spheres. The intersection test between two medial primitives is formulated as a quadratically constrained quadratic program problem. We give an algorithm to solve this problem exactly, which returns the deepest penetration between a pair of intersecting medial primitives. When coupled with spatial hashing, collision (including self-collision) can be efficiently identified on GPU within few milliseconds even for massive simulations. We have tested our system on a variety of geometrically complex and high-resolution deformable objects, and our system produces convincing animations with all the collisions/self-collisions well handled at an interactive rate.

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

Additional Key Words and Phrases: Medial axis, Deformable model, Collision culling, GPU

ACM Reference Format:
Anonymous Author(s). 2019. Medial Elastics: Efficient and Collision-ready Deformation via Medial Axis Transform. *ACM Trans. Graph.* 1, 1 (October 2019), 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

0730-0301/2019/10-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

A high-quality nonlinear deformable simulation of elastic bodies is an important procedure for physics animations. In order to calculate the time-dependent displacement of a deformable object, one needs to solve a nonlinear system of the dynamic equilibrium. The simulator should sufficiently lower the system residual to avoid visual artifacts and minimize the numerical instability along the time integration. For real-time animation with a small time budget, it is common to use coarsened or reduced simulation techniques to formulate the dynamics with a generalized coordinate instead of the per-vertex displacement vector [Sifakis and Barbic 2012]. As a result, the core simulation is independent of the resolution of the input model. Meanwhile, collision detection (CD) including self-collision detection (SCD) is another essential task along the animation pipeline, which returns all the overlapping triangle pairs on deformed surfaces of elastic bodies so that interpenetrating objects can be resolved using complimentary constraints or penalty forces [Baraff 1994; Moore and Wilhelms 1988]. Nowadays,

models with hundred of thousands or even millions of surface triangles are ubiquitous, and a brute-force pair-wise triangle intersection test is clearly infeasible. Collision and self-collision culling (CC/SCC) [Bergen 1997; Bridson et al. 2002; Gottschalk et al. 1996] pre-screens the updated geometry of models, precludes triangles that do not collide, and prevents us from testing exceeding pairs.

Both deformable simulation and collision culling have been extensively studied in the graphics community, yet they are normally considered as two independent problems and handled separately: the simulator calculates the displacement of the deformable body by updating the generalized coordinates. The new surface geometry is then extracted and forwarded as the input for the follow-up CC/SCC algorithms. This system flow overlooks the fact that *both reduced simulation and collision culling rely on good geometric approximations of dynamically deformed objects* and breaks the possible algorithmic connection between them, which can potentially improve the performance of the animation pipeline as a whole. Observing this limitation, James and Pai [2004] exploited subspace modes to construct a bounded deformation tree (BD-tree) for fast CC without referring to the fullspace displacement. Barbič and James [2010] accelerated the SCC by pre-computing subspace SCC certificates, which demarcate a proven self-collision free area on the model with reduced coordinates. However, these methods tend to work with an arbitrary reduced model and give up the opportunity of fine-tuning the subspace construction to maximize CC/SCC performance.

In this paper, we propose a new framework that unifies the reduced deformable simulation and CC/SCC by leveraging the *medial axis transform* (MAT) and its corresponding *medial mesh* (MM) of an input deformable body. The *medial axis* (MA) of a 3D model consists of a set of points that are centers of maximally inscribed spheres (i.e. the medial spheres) touching the boundary with at least two contacts. MM is a non-manifold triangle mesh that discretizes MA with piecewise linear segments. MAT houses the information of both MM and radii of all the medial spheres at MM vertices. It essentially forms a collection of linearly interpolated spheres or *medial primitives* (MP). MAT has long been considered as a fundamental volumetric descriptor of 3D shapes [Faraj et al. 2013; Sun et al. 2016] and an effective shape approximation representation [Stolpner et al. 2012; Yang et al. 2018]. If we slightly dilate the radii of MPs on the MM, a high-quality tight enclosure of the input model can be obtained. Inspired by these unique properties of MAT and MM, we design our new framework, named Medial Elastics, with the following noteworthy technical features:

- We adopt a spatial reduction scheme based on the embedded MM of a deformable object. We assign each medial vertex a handle, holding certain transformation freedoms to drive the deformation of the 3D model. MM has long been regarded as the "skeleton" of a 3D shape, which naturally characterizes its most dominant nonlinear dynamics. Therefore, our subspace is expressive and compact with well-preserved local details.
- We follow the local-global alternating strategy of the projective dynamics [Bouaziz et al. 2014] to solve our reduced model. Within this context, we carefully assess the benefit of applying model reduction on global and local steps respectively, trying to obtain an optimal trade-off between the computation efficiency and simulation quality. We find that an aggressive reduction at the local step could severely downgrade the plausibility of the animation, yet with only marginal performance speed up. Therefore, we design a semi-reduced projective dynamics simulator that only applies the subspace projection at the global step. Compared with a full-reduction scheme (e.g. the hyper-reduced projective dynamics solver [Brandt et al. 2018]), our framework delivers richer deformation effects which in many cases, are visually indistinguishable with the full simulation.
- In our framework, CC and SCC start directly with the generalized coordinate i.e. the state vector of the MM, which fully determines the configuration of the MAT. As MAT is able to tightly encapsulate a deformed model using only a few MPs, MAT-based collision culling is much more effective and efficient compared with the state-of-the-arts. We not only give an efficient MAT-based CC algorithm, but also derive a closed-form formulation computing the deepest interpenetration between colliding MPs. This allows us to directly query for intersecting triangles nearby the deepest penetrations among very few MP candidates for CD and SCD.

We have tested our method with a variety of complex high-resolution 3D models and challenging simulation scenarios. As reported in Fig. 1, our system yields high-quality collision-free deformable animations with rich local details at an interactive rate.

2 RELATED WORK

An efficient and plausible simulation of elastically deformable objects is a desired feature for many applications. In general, deformable simulation is formulated as a dynamic equilibrium. For high-resolution models, the simulator needs to solve a large-scale nonlinear system repeatedly at each time step. Therefore, even if we have many well-established simulation frameworks such as the finite element method (FEM) [Sifakis and Barbic 2012], finite difference method [Zhu et al. 2010], meshless method [Martin et al. 2010; Müller et al. 2005], mass-spring system [Liu et al. 2013], and material particle method [Gao et al. 2017], etc., efficiently simulating high-resolution models remains a challenging problem.

Simulation speedup can be achieved using model reduction, which removes less important degrees of freedom (DOFs) and creates a subspace representation of fullspace DOFs. *Spectral reduction* method like modal analysis [Choi and Ko 2005; Hauser et al. 2003; Pentland and Williams 1989] and its first-order modal derivatives [Barbič and James 2005; Yang et al. 2015] are often considered as the most effective way for the spectral subspace construction. Displacement vectors from recent fullspace simulations can also be utilized as subspace bases [Kim and James 2009]. Another collection of research that shares a similar idea of reducing the total number of simulation DOFs is referred to as *spatial reduction* method in this paper. Spatial reduction leverages coarsened geometry data structures to prescribe the dynamics of a fine model. The most famous paradigm of spatial reduction in graphics may be the animation skinning, where the deformed skin is attached and controlled by DOFs at rigid joints [Kry et al. 2002]. Likewise, Capell et al. [2002] deformed an elastic body using an embedded skeleton; Gilles et al. [2011] used 6-DOF rigid frames to drive the deformable simulation; Faure et al. [2011] used scattered *handles* to model complex deformable models; Martin et al.

[2010] used sparsely distributed integrators named *elastons* to model the nonlinear dynamics of rod, shell, and solid uniformly.

Our simulation framework is based on *projective dynamics* [Bouaziz et al. 2014]. This method decomposes the numerical system into many small local constraint projections that can be processed in parallel, and then reassembled in a non-linear fashion leading to accurate global dynamics. A number of approaches have been proposed to accelerate its computation by using various iterative GPU solvers [Fratarcangeli et al. 2016; Narain et al. 2016; Wang 2015; Wang and Yang 2016]. Brandt et al. [2018] combined projective dynamics with spatial reduction to simulate high-resolution models in real time. We are inspired by excellent efficiency of this method. However, we also found that the accuracy of the local projection is critical to the animation quality, and local projection should not be over-aggressively reduced.

CD/SCD is another important procedure along the physics animation pipeline. A commonly adopted method is to use some bounding volume hierarchy (BVH) [Zachmann and Langetepe 2003] to avoid excessive triangle-triangle intersection tests. Various BVH types have been explored such as AABB [Bergen 1997], OBB [Gottschalk et al. 1996], sphere [Hubbard 1995; James and Pai 2004], Boxtree [Zachmann 2002], spherical shell [Krishnan et al. 1998] and so on. Most existing BVHs are built with *fixed bounding primitives* like spheres or boxes. We say they are fixed meaning each primitive is nailed at a prescribed position, enveloping a small surrounding volume. Therefore, it often needs a large number of such fixed primitives to envelop a complex shape. For rigid body animation, updating a BVH is efficient as the entire BVH can be uniformly rotated and translated with six rigid body DOFs. However, updating BVH for a deformable object is, in general, expensive, which often takes a time at the order of $\mathbf{O}(N)$, where N denotes the model size [Teschner et al. 2005; Wang et al. 2018]. Kavan and Zara [2005] speeded up BVH update leveraging the displacement convexity of linear blend skinning (LBS) algorithm. Our method uses MAT as the bounding volume, which contains infinitely many linearly interpolated spheres along the underlying MM. There exist other data structures like the *sphere mesh* [Thiery et al. 2013, 2016; Tkach et al. 2016] that also use linearly interpolated spheres. However, only MAT guarantees that spheres are maximally inscribed in the local surface. As a result, CC and SCC become much more effective with MAT.

We are not the first trying to couple model reduction and CC/SCC. James and Pai [2004] proposed an algorithm that updates the BD-tree directly using the generalized coordinate. This method was later generalized to enable efficient Haptics force rendering [Barbič and James 2007]. Barbič and James [2010] computed SCC *certificates* in subspace to accelerate SCC. Based on the observation that a self-collision occurs under large local deformation, Zheng and James [2012] proposed an energy-based metric to improve the effectiveness of SCC. Teng et al. [2014] leveraged the coherence of the joint contact in character animation, and efficiently processed the collision within the subspace. These contributions intend to improve the performance of CC/SCC for any reduced models. The work from Zheng and James [2012] is more general, being applicable to arbitrary deformations of triangle meshes even without reduction. However, generalized coordinates in different reduced models have distinct geometry or physics interpretations. It is unlikely that a single framework can fully leverage all the features of various reduced models. A recent contribution similar to our method is VIPER [Angles et al. 2019], which was built by adding scaling DOFs at the cross-section of the classic cosserat model. VIPER handles collision using iterative Dichotomous search [Antoniou and Lu 2007].

We present a new framework namely *Medial Elastics*. Our framework seamlessly integrates model reduction and CC/SCC under a unique MAT-based simulation scheme. From model reduction perspective, MAT naturally captures the most important deformations of an elastic object, and allows to construct a compact yet expressive subspace. We adopt a semi-reduced projective dynamics formulation to better balance the trade-off between efficiency and quality. This formulation well synergizes with our collision-ready strategy, which makes the reduced global matrix collision-invariant and prefactorizable. The reduced coordinate is directly used to update the bounding MAT at the cost of $\mathbf{O}(n)$, where $n << N$ is the subspace dimension. From CC/SCC point of view, MAT tightly envelops the deformable body with only a few hundred MPs. We give the analytic solution to calculate the deepest intersection between two MPs. As a result, our MAT-based subspace CC/SCC only needs $\mathbf{O}(n^2)$ calculations for a pair of deformable objects, which is also parallelizable on the GPU.

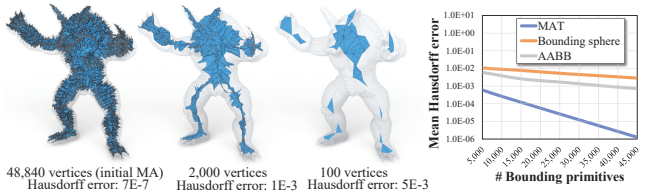

Fig. 2. Volumetric MAT approximation of the Armadillo model (with 62, 161 surface triangles). The initial MA (the leftmost subfigure) contains around $50k$ medial vertices. After aggressively simplifying the MM to only 100 vertices, the mean Hausdorff error remains less than 1%. Note that the vertical axis of the rightmost plot is logarithmic.

3 MEDIAL AXIS TRANSFORM

As shown in Fig. 2, the MA of 3D model is a set of centers of maximally inscribed spheres with at least two closest points on its boundary [Blum 1967]. By incorporating the radius information along the MA, there exists a *unique* MAT representing the original 3D model losslessly. One can easily perform the in/out test by comparing the distance to the nearest medial point and the corresponding radius. Sun et al. [2016] showed that MA can be compactly approximated by MM, a 2D non-manifold triangle mesh, and the radius information is interpolated among two (across an edge) or three (across a triangle) medial vertices on the MM. The resulting MP is either a medial cone or a medial slab (i.e. see Fig. 3). After simplification, the MAT is no longer fully consistent with the original

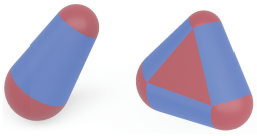

Fig. 3. Linearly interpolating medial spheres over an edge or a triangle yields a medial cone (left) or a medial slab (right).

model. Nevertheless, the quality of MAT is insensitive to the MM simplification. As shown in Fig. 2, even with an aggressive simplification removing 99.8% medial vertices (from 48,840 to 100), the mean Hausdorff error of the resulting MAT is still below 1%. Here, the so-called *Hausdorff error* refers to the Hausdorff distance divided by the maximum dimension of the model's bounding box.

An initial MA is obtained from a filtered volumetric Voronoi tessellation of the input model [Amenta and Bern 1999; Attali and Montanvert 1997]. This MA is normally noisy and has many spiky edges (i.e. the leftmost subfigure in Fig. 2). We follow the greedy algorithm as described in [Faraj et al. 2013; Li et al. 2015] to obtain the MM using edge-collapse. In our implementation, we use the quadratic error metric as in Q-MAT [Li et al. 2015] for the MM simplification, which effectively removes redundant and noisy spikes on the initial MM. After the MM simplification is completed, we need to ensure that all the surface vertices of the input model are within the MAT volume. This is achieved by *locally* scaling radii of MPs for "most outside" vertices so that they are just enclosed by the scaled spheres.

Unlike other widely used bounding primitives like AABB [Bergen 1997], OBB [Gottschalk et al. 1996] and bounding spheres [Hubbard 1995; James and Pai 2004], MAT enables the linear variation along edges and triangles on the MM, which injects one more dimension of approximation freedom. Consequently, MAT is also one-order more effective in terms of the volumetric shape approximation. This is verified in the rightmost plot in Fig. 2, where we compare mean Hausdorff errors using three different bounding primitives: AABB, bounding sphere, and MAT to encapsulate the Armadillo model. It can be seen from the plot that with the same number of bounding primitives, MAT consistently has a much smaller Hausdorff error than both AABB and bounding sphere. In CD, one needs to pinpoint an intersecting triangle pair between two geometries, which is quadratic w.r.t. the number of participating bounding primitives. For deformable models with complex and irregular surface geometries (like many examples shown in the paper), the advantage of MAT is even more significant. Therefore, it is not surprising to see that MAT-based CC/SCC is orders-of-magnitude more effective than existing bounding primitives.

4 MAT-DRIVEN SEMI-REDUCED PROJECTIVE DYNAMICS

Regarding deformable simulation, our method can be considered as a spatial reduction technique, which prescribes the nonlinear dynamic of an elastic body with a collection of sparsely distributed *handles*. Those handles are representatives of simulation DOFs, and the generalized coordinate $\mathbf{q}$ is essentially the state vector of all the handles. MA has been known to be the generic "topological skeleton" of a 3D shape suggesting its expressivity for characterizing intrinsic deformations of 3D shapes. We follow this intuition and allocate handles at medial vertices on the MM, which will be referred to as *medial handles* (MHs).

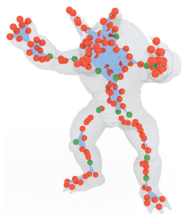

Fig. 4. Affine (red) and quadratic (green) handles on the Armadillo model.

4.1 Subspace Construction

Given a material point p, let $\mathbf{x} = [x_1, x_2, x_3]^\top$ and $\mathbf{u} = [u_1, u_2, u_3]^\top$ be its rest-shape position and displacement. We model its per-component displacement from an MH with a transformation function T_i such that: $u_i = T_i(\mathbf{x})$ for $i = 1, 2, 3$. A common choice of T is the affine transformation [Faure et al. 2011] or the rigid body transformation [Gilles et al. 2011]. For instance, if we set an MH as an affine handle, $T_i = \mathbf{a}_i \cdot \mathbf{x} + t_i$ or $\mathbf{u} = \mathbf{A}\mathbf{x} + \mathbf{t}$, where $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3]^\top$ is the corresponding linear transformation matrix and $\mathbf{t} = [t_1, t_2, t_3]^\top$ are three translational freedoms. A fundamental limitation of using affine/linear DOFs in spatial reduction is the locking issue: with coarsened discretization, the piecewise linear approximation often yields a high shearing energy[1], which prevents the elastic body from being properly deformed to the desired shape, and incorporating high-order DOFs is recommended [Bargteil and Cohen 2014; Martin et al. 2010]. Therefore, in addition to conventional affine MHs, we also have a few quadratic MHs on the MM to provide sufficient flexibility for large, nonlinear and local deformations (Fig. 4).

If a medial vertex is connected with very few other handles, its local discretization may be too sparse, and a quadratic handle is assigned. We also downgrade quadratic handles that merely cover a small volume on the deformable body to be an affine one. The threshold is set as $\frac{1}{10} \cdot \frac{V}{n}$, where V is the volume of the entire deformable body, and n is total number of MHs. Given a quadratic MH $\mathcal{H}_j$, we follow the formulation in [Luo et al. 2018] and define T_i^j as a quadratic form of: $T_i^j(\mathbf{x}) = \mathbf{x}^\top \mathbf{Q}_i^j \mathbf{x} + \mathbf{a}_i^{j\top} \mathbf{x} + t_i^j$, where $\mathbf{Q}_i^j$ is a 3-by-3 symmetric tensor encoding quadratic deformation freedoms. The reduced coordinate of $\mathcal{H}_j$ can then be written as:

$$\mathbf{q}^j = \left[\mathbf{t}^{j\top}, \mathbf{a}_1^{j\top}, \mathbf{a}_2^{j\top}, \mathbf{a}_3^{j\top}, \widetilde{\mathbf{q}}_1^{j\top}, \widetilde{\mathbf{q}}_2^{j\top}, \widetilde{\mathbf{q}}_3^{j\top}, \widehat{\mathbf{q}}_1^{j\top}, \widehat{\mathbf{q}}_2^{j\top}, \widehat{\mathbf{q}}_3^{j\top}\right]^\top \in \mathbb{R}^{30}. \quad (1)$$

Here, we use superscript $(\cdot)^j$ to denote the handle index. Diagonal and off-diagonal DOFs of $\mathbf{Q}_i^j$ are vectorized as:

$$\widetilde{\mathbf{q}}_i^j = \left[(\mathbf{Q}_i^j)_{11}, (\mathbf{Q}_i^j)_{22}, (\mathbf{Q}_i^j)_{33}\right]^\top, \text{and } \widehat{\mathbf{q}}_i^j = 2\left[(\mathbf{Q}_i^j)_{12}, (\mathbf{Q}_i^j)_{23}, (\mathbf{Q}_i^j)_{13}\right]^\top.$$

The subspace matrix $\mathbf{U}$ at p relates $\mathbf{q}^j$ to its displacement via $\mathbf{u} = \mathbf{U}\mathbf{q}^j$, and it has the following structure:

$$\mathbf{U}(\mathbf{x}) = \left[\mathbf{U}_t, \mathbf{U}_a, \widetilde{\mathbf{U}}, \widehat{\mathbf{U}}\right] = \left[\mathbf{I}, \mathbf{I} \otimes \mathbf{x}^\top, \mathbf{I} \otimes \widetilde{\mathbf{x}}^\top, \mathbf{I} \otimes \widehat{\mathbf{x}}^\top\right] \in \mathbb{R}^{3\times 30}, \quad (2)$$

with $\widetilde{\mathbf{x}} = [x_1^2, x_2^2, x_3^2]^\top$ and $\widehat{\mathbf{x}} = [x_1x_2, x_2x_3, x_1x_3]^\top$ being second-order homogenous and heterogenous position vectors of p. The final displacement of p is computed by assembling a global subspace matrix and reduced coordinate that blend T^j of all the n handles:

$$\mathbf{u} = \mathbf{U}\mathbf{q} = \sum_{j=1}^{n} w_j \mathbf{U}(\mathbf{x})\mathbf{q}^j = [w_1\mathbf{U}(\mathbf{x}), \cdots, w_n\mathbf{U}(\mathbf{x})] \begin{bmatrix} \mathbf{q}^1 \\ \vdots \\ \mathbf{q}^n \end{bmatrix}. \quad (3)$$

The weight coefficient $w_j(\mathbf{x})$ is position-dependent and varies at different vertices on the mesh. Yet, it can be pre-computed and remains unchanged during the simulation. We use the biharmonic weight [Jacobson et al. 2011] to assign w_j for each MH at a given mesh vertex by solving the mass-weighted Laplacian matrix. During

[1]This is because shearing is the first-order approximation of many nonlinear effects like bending and twisting.

the weight computation, we impose the interpolating constraint of: $w_j(MH_j) = 1$ and $w_j(MH_i) = 0$ for $i \neq j$. In other words, each handle is solely controlled by its own generalized coordinate. If there exist extra boundary conditions of the deformable body (i.e. anchor vertices), the weight coefficients at those vertices are also constrained to be zero so that the boundary conditions can be automatically satisfied without imposing additional constraints during the simulation. As the biharmonic weight itself has good locality, we do not explicitly tune the supporting radius of the weight function as did in [Brandt et al. 2018] in general. If the deformable object uses a highly heterogenous material, one should choose a more material-aware weight mechanism such as in [Luo et al. 2018] or [Nesme et al. 2009].

4.2 Semi-reduced Projective Dynamics

With the implicit Euler integration scheme, the dynamic equilibrium of the deformable body becomes:

$$\mathbf{M}(\mathbf{u}_{t+1} - \mathbf{u}_t - h\dot{\mathbf{u}}_t) = h^2\big(\mathbf{f}_{\text{int}}(\mathbf{u}_{t+1}) + \mathbf{f}_{\text{ext}}\big), \tag{4}$$

where $\mathbf{M}$ is the mass matrix. $\mathbf{f}_{\text{int}}$ and $\mathbf{f}_{\text{ext}}$ stand for the elastic internal force and the external force. The subscript $(\cdot)_t$ denotes the time integration step, and h is the time step size. Projective dynamics poses the equilibrium of Eq. (4) as an optimization problem of $\mathbf{u}_{t+1} = \arg\min E(\mathbf{u})$ for

$$E(\mathbf{u}) = \frac{1}{2h^2}\left\|\mathbf{M}^{\frac{1}{2}}(\mathbf{u} - \mathbf{u}^*)\right\|^2 + \sum W(\mathbf{u}), \tag{5}$$

where $\mathbf{u}^* = \mathbf{u}_t + h\dot{\mathbf{u}}_t + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$ is a known vector depending on the kinematics from the previous time step. W is the constraint energy measuring how far the system deviates from its nearest optimal configuration, where all the constraints are satisfied. It is solved via an efficient local-global alternating procedure utilizing an auxiliary vector $\mathbf{p}$. At the local step, projective dynamics takes a given mesh deformation $\mathbf{u}$ and seeks for the optimal configuration of $\mathbf{p}_i$ for the i-th constraint, which represents an abstract point on the constraint manifold:

$$\min_{\mathbf{p}_i} \frac{\omega_i}{2}\|\mathbf{A}_i\mathbf{S}_i\mathbf{u} - \mathbf{B}_i\mathbf{p}_i\|^2 \quad \text{s.t.} \quad C_i(\mathbf{p}_i) = 0, \tag{6}$$

where ω_i is the weight of the i-th constraint. $\mathbf{S}_i$ is a selection matrix that extracts fullspace freedoms in $\mathbf{u}$ associated with the i-th constraint. $\mathbf{A}_i$ and $\mathbf{B}_i$ are constant matrices typically encoding certain linear operators that lead to a suitable distance measure between $\mathbf{u}$ and $\mathbf{p}$. $C_i(\mathbf{p}_i) = 0$ defines the constraint manifold where $\mathbf{p}_i$ resides. After the local step, the global step fixes all the $\mathbf{p}_i$ and computes $\mathbf{u}$ via solving the linear system of:

$$\left(\frac{\mathbf{M}}{h^2} + \sum_i \omega_i\mathbf{S}_i^\top\mathbf{A}_i^\top\mathbf{A}_i\mathbf{S}_i\right)\mathbf{u} = \frac{\mathbf{M}}{h^2}\mathbf{u}^* + \sum \omega_i\mathbf{S}_i^\top\mathbf{A}_i^\top\mathbf{B}_i\mathbf{p}_i. \tag{7}$$

Global reduction vs. local reduction We argue that *performing model reduction on global and local steps is not equally profitable.* The global reduction (i.e. use model reduction at the global step) accelerates the simulation more substantially and induces less compromise to the animation quality than the local reduction. To show this, we first look into the acceleration potential of global and local reduction respectively. In the local step, each constraint C_i is normally of low dimension, and finding $\mathbf{p}_i$ for an individual constraint (namely, the constraint projection) is considered as an $\mathbf{O}(1)$ procedure. With the help of GPU parallelization, the cost of the local step is sublinear w.r.t. the problem size. As plotted in Fig. 6, the `nVidia Titan` GPU only needs less than 1.1 *ms* to handle 500, 000 constraints. At the global step, we solve a constant linear system of $\frac{\mathbf{M}}{h^2} + \sum_i \omega_i\mathbf{S}_i^\top\mathbf{A}_i^\top\mathbf{A}_i\mathbf{S}_i$ using the forward and backward substitution. This procedure is sequential. While it can still be accelerated by the GPU as in [Naumov 2011], the time complexity remains quadratic. In other words, the global reduction improves the time performance more significantly from $\mathbf{O}(N^2)$ to $\mathbf{O}(n^2)$ compared to the $\mathbf{O}(N)$ to $\mathbf{O}(n)$ acceleration at the local step (also see Fig. 6). We are aware of the possibility of using iterative linear solvers over the direct solver [Fratarcangeli et al. 2016, 2018; Wang 2015; Wang and Yang 2016] for the global step. However, those methods are only effective for sparse systems. With model reduction, the sparsity of the global matrix is highly condensed, and those acceleration techniques become less profitable.

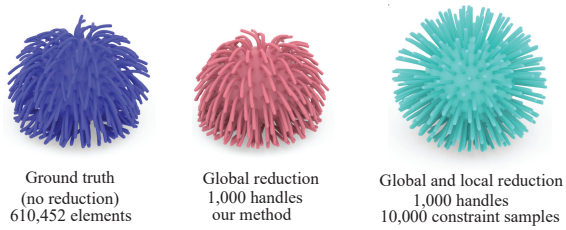

Fig. 5. The puffer ball has sharp concave local geometry, which is typically considered less friendly to model reduction. Nevertheless, our method (with only global reduction) yields plausible animation, that is visually indistinguishable from the fullspace simulation using only 1, 000 MHs. However, if local reduction is also applied as in [Brandt et al. 2018], we can clearly see the stiffening artifacts from the animation with pale local effects.

Next, we investigate the quality of the resulting animation when model reduction is applied to local and global steps. It is true that the subspace of $\mathbf{u} = \mathbf{U}\mathbf{q}$ imposes a hard constraint on all the possible deformations we could obtain. Yet, as long as we have a modest-size MM (i.e. with hundreds of handles) this subspace often contains copious interesting effects to produce a compelling animation. The real question is whether the solver is able to reach desired (subspace) deformations if the local step is also reduced. With global reduction alone, the solver is still able to find the optimal configuration for each constraint, which most effectively reduces system's residual error (locally). The primary compromise global reduction takes is at the distance measure – a fullspace local minimizer of Eq. (5), after finishing all the constraint projections, is approximated

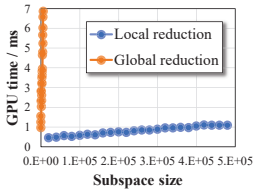

Fig. 6. The local step is sublinear w.r.t. subspace dimension. Projecting 500, 000 as-rigid-as-possible constraints only takes 1.1 *ms* on the `Titan` GPU. Global step needs to solve a linear system and is more sensitive to the problem size.

by its best subspace replica. On the other hand, local reduction directly smooths the constraint manifold C_i, which is often nonlinear and concave. All the approximated constraint projections induce residual errors to the system. Some of them may be filtered by the subspace projection but most of them are accumulated due to the nonlinear nature of constraints. Therefore, local reduction tends to produce more artifacts, and should be used with extra cautions.

This analysis is consistent with our observation from the experiment. As shown in Fig. 5, we simulate a soft puffer ball using full projective dynamics, semi-reduced projective dynamics (our method, global reduction only), and hyper-reduced projective dynamics (both global and local steps are reduced as in [Brandt et al. 2018]). Both semi-reduced and hyper-reduced methods use the same subspace constructed from 1,000 MHs. We can see from the figure that our semi-reduction scheme yields a similar result as the fullspace projective dynamics. However, hyper-reduced projective dynamics produces stiffening artifacts: most strings on the sphere do not deform properly under the gravity. In this example, each solver is fully converged. We use 10,000 constraint samples for the hyper-reduced case – this number is much higher than constraint samples reported in [Brandt et al. 2018].

Bearing the above analysis in mind, our semi-reduced projective dynamics scheme is simply to solve Eq. (7) within the column space of $\mathbf{U}$ while enforcing $\mathbf{u} = \mathbf{U}\mathbf{q}$:

$$\mathbf{U}^\top \left(\frac{\mathbf{M}}{h^2} + \sum_i \omega_i \mathbf{S}_i^\top \mathbf{A}_i^\top \mathbf{A}_i \mathbf{S}_i \right) \mathbf{U}\mathbf{q} = \mathbf{U}^\top \left(\frac{\mathbf{M}}{h^2} \mathbf{U}\mathbf{q}^* + \sum \omega_i \mathbf{S}_i^\top \mathbf{A}_i^\top \mathbf{B}_i \mathbf{p}_i \right). \tag{8}$$

Note that $\mathbf{q}^*$ should be computed via $\mathbf{q}_t + h\dot{\mathbf{q}}_t + h^2 \left(\mathbf{U}^\top \mathbf{M}\mathbf{U}\right)^{-1} \mathbf{U}^\top \mathbf{f}_{\text{ext}}$ instead of $\mathbf{q}_t + h\dot{\mathbf{q}}_t + h^2 \mathbf{U}^\top \mathbf{M}^{-1} \mathbf{f}_{\text{ext}}$. This is because subspace bases are not orthonormal as eigen modes, such as column vectors in $\mathbf{U}$. If a Gram-Schmidt orthogonalization is performed over them explicitly, the geometric interpretation of generalized coordinate may be destroyed. For local step, we leave it untouched in the fullspace and only pre-compute $\left(\mathbf{A}_i \mathbf{S}_i \mathbf{U}\right)^\top \mathbf{B}_i$ and $\mathbf{A}_i \mathbf{S}_i \mathbf{U}$ for each constraint.

4.3 Making Subspace Collision-Ready

In projective dynamics, collisions can also be regarded as a type of constraint. However, unlike other constraint types e.g. volume preservation or elastic potential, collision constraints are dynamically changing during the animation. In other words, the reduced global matrix is no longer constant and cannot be pre-factorized unless another external collision handling method is used (e.g. the penalty method). This limitation can be removed within our semi-reduced formulation. The key observation is that the collision constraint does not alter the configuration of the subspace. It only induces the auxiliary variable $\mathbf{p}_i$ for each collision point at the local step, which is in the fullspace. Therefore, our strategy is to put all the surface vertices into a "ghost collision event" before the global matrix assembly. Each surface vertex will be associated with a virtual constraint, and $\mathbf{A}_i$ and $\mathbf{S}_i$ matrices are included in the reduced global matrix. If the vertex is collision-free, the constraint projection simply setting the optimal position as its current position. The constraint projection only yields an acting $\mathbf{p}_i$ when the vertex truly collides with something. Intuitively, this strategy is similar to pre-attaching each surface vertex a virtual spring. The spring does not yield penalty forces until a collision is detected. On the downside, this scheme impairs the convergency rate when massive collisions become active as the constraint manifold would be significantly altered within a time step. A possible cure of this limitation is to use multiple global matrices as in [Komaritzan and Botsch 2018]. Clearly, doing so requires extra GPU memory consumption, which could be prohibitive for massive simulations.

4.4 Displacement and Deformation Bounding

Before the simulation, the rest-shape MAT tightly encapsulates the undeformed model making it an ideal bounding envelope for CC/SCC. After simulation starts, we adjust states of all the medial spheres at each time step so that the updated MAT also well encapsulates the deformed model, and the cost of updating the MAT is of the reduced order of $\mathbf{O}(n)$.

Displacement bounding Let us consider a vertex within a medial cone whose rest position is $\mathbf{x}$. There exists a sphere $\mathcal{S}$ centered at $\mathbf{c}$ encapsulating the vertex such that: $\|\mathbf{x}-\mathbf{c}\| < r$, where r is the radius of $\mathcal{S}$. We know that $\mathcal{S}$ is actually interpolated by medial spheres at two handles, say $\mathcal{H}_1$ and $\mathcal{H}_2$, of the cone, whose centers and radii are $\mathbf{c}_1$, $\mathbf{c}_2$ and r_1, r_2 respectively. In other words, we have $r = t_1 r_1 + t_2 r_2$ and $\mathbf{c} = t_1 \mathbf{c}_1 + t_2 \mathbf{c}_2$ with interpolating parameters t_1 and t_2 satisfying $t_1 + t_2 = 1$ and $0 \le t_1, t_2 \le 1$. During the simulation when the vertex is displaced by $\mathbf{u}$, the distance between the deformed vertex and the sphere center becomes $\|\mathbf{x} + \mathbf{u} - \mathbf{c}'\|$. While the center of $\mathcal{S}$ is also moved from $\mathbf{c}$ to $\mathbf{c}'$ due to the deformation, it is still the interpolation of two corresponding medial spheres. Let $\mathbf{u}_1$ and $\mathbf{u}_2$ be the handle displacements, and we can re-write $\mathbf{x} + \mathbf{u} - \mathbf{c}'$ as:

$$\begin{aligned} \mathbf{x} + \mathbf{u} - \mathbf{c}' &= \mathbf{x} + \mathbf{u} - \left[t_1(\mathbf{c}_1 + \mathbf{u}_1) + t_2(\mathbf{c}_2 + \mathbf{u}_2) \right] \\ &= \mathbf{x} - \mathbf{c} + \mathbf{u} - t_1 \mathbf{u}_1 - t_2 \mathbf{u}_2. \end{aligned} \tag{9}$$

Let us first assume that $\mathcal{H}_1$ and $\mathcal{H}_2$ are the only handles on the MM implying $\mathbf{u} = w_1 T_1(\mathbf{x}) + w_2 T_2(\mathbf{x})$. We also have $\mathbf{u}_1 = T_1(\mathbf{c}_1)$ and $\mathbf{u}_2 = T_2(\mathbf{c}_2)$ because $\mathbf{c}_1$ and $\mathbf{c}_2$ are actually rest-shape positions of $\mathcal{H}_1$ and $\mathcal{H}_2$. Following the triangle inequality, an upper bound of the new distance can be obtained as:

$$\left\|\mathbf{x} + \mathbf{u} - \mathbf{c}'\right\| = \|\mathbf{x} - \mathbf{c} + \Delta\mathbf{u}_1 + \Delta\mathbf{u}_2\| \le r + \|\Delta\mathbf{u}_1\| + \|\Delta\mathbf{u}_2\|, \tag{10}$$

where $\Delta\mathbf{u}_1 = w_1 T_1(\mathbf{x}) - t_1 T_1(\mathbf{c}_1)$ and $\Delta\mathbf{u}_2 = w_2 T_2(\mathbf{x}) - t_2 T_2(\mathbf{c}_2)$. We further decompose $\Delta\mathbf{u}_1$ into four components:

$$\begin{aligned} \Delta\mathbf{u}_1 = \; & \left(w_1 \mathbf{U}_t(\mathbf{x}) - t_1 \mathbf{U}_t(\mathbf{c}_1)\right)\mathbf{t}^1 + \left(w_1 \mathbf{U}_a(\mathbf{x}) - t_1 \mathbf{U}_a(\mathbf{c}_1)\right)\mathbf{a}^1 \\ & + \left(w_1 \widetilde{\mathbf{U}}(\mathbf{x}) - t_1 \widetilde{\mathbf{U}}(\mathbf{c}_1)\right)\widetilde{\mathbf{q}}^1 + \left(w_1 \widehat{\mathbf{U}}(\mathbf{x}) - t_1 \widehat{\mathbf{U}}(\mathbf{c}_1)\right)\widehat{\mathbf{q}}^1. \end{aligned} \tag{11}$$

It is easy to see as per Eqs. (1) and (2) that $\mathbf{t}^1$, $\mathbf{a}^1 = \left[\mathbf{a}_1^{1\top}, \mathbf{a}_2^{1\top}, \mathbf{a}_3^{1\top}\right]^\top$, $\widetilde{\mathbf{q}}^1 = \left[\widetilde{\mathbf{q}}_1^{1\top}, \widetilde{\mathbf{q}}_2^{1\top}, \widetilde{\mathbf{q}}_3^{1\top}\right]^\top$, and $\widehat{\mathbf{q}}^1 = \left[\widehat{\mathbf{q}}_1^{1\top}, \widehat{\mathbf{q}}_2^{1\top}, \widehat{\mathbf{q}}_3^{1\top}\right]^\top$ are generalized coordinates for translation, affine, homogenous quadratic, and heterogenous quadratic DOFs of $\mathcal{H}_1$. Next, we derive an upper bound of $\|\Delta\mathbf{u}_1\|$ by finding upper bounds of the norm of each of these four displacement components.

The translation component is relatively simple. Eq. (2) tells that $\mathbf{U}_t(\mathbf{x}) = \mathbf{U}_t(\mathbf{c}_1) = \mathbf{I}$ are constant, which yields:

$$\begin{aligned} \left\|\left(w_1 \mathbf{U}_t(\mathbf{x}) - t_1 \mathbf{U}_t(\mathbf{c}_1)\right)\mathbf{t}^1\right\| &= \left\|(w_1 - t_1) \cdot \mathbf{t}^1\right\| \\ &\le \max\left\{|w_1 - t_1|\right\} \cdot \left\|\mathbf{t}^1\right\|. \end{aligned} \tag{12}$$

In other words, when the simulator returns a reduced displacement at a certain time step, $\mathbf{t}^1$ is given, and the maximum length of the translation displacement is capped by the maximum value of $|w_1 - t_1|$, which can be pre-computed before simulation.

$\mathbf{U}_a(\mathbf{x})$ is not constant, and we re-write the affine displacement by substituting $\mathbf{U}_a(\mathbf{x}) = \mathbf{I} \otimes \mathbf{x}^\top$ as:

$$\begin{aligned}\big(w_1\mathbf{U}_a(\mathbf{x}) - t_1\mathbf{U}_a(\mathbf{c}_1)\big)\mathbf{a}^1 &= \left[w_1(\mathbf{I}\otimes\mathbf{x}^\top) - t_1(\mathbf{I}\otimes\mathbf{c}_1^\top)\right]\mathbf{a}^1 \\ &= \left[\mathbf{I}\otimes(w_1\mathbf{x} - t_1\mathbf{c}_1)^\top\right]\mathbf{a}^1,\end{aligned} \tag{13}$$

which leads us to:

$$\begin{aligned}&\left\|\big(w_1\mathbf{U}_a(\mathbf{x}) - t_1\mathbf{U}_a(\mathbf{c}_1)\big)\mathbf{a}^1\right\| = \left\|\left[\mathbf{I}\otimes(w_1\mathbf{x} - t_1\mathbf{c}_1)^\top\right]\mathbf{a}^1\right\| \\ &= \sqrt{\big((w_1\mathbf{x} - t_1\mathbf{c}_1)\cdot\mathbf{a}_1^1\big)^2 + \big((w_1\mathbf{x} - t_1\mathbf{c}_1)\cdot\mathbf{a}_2^1\big)^2 + \big((w_1\mathbf{x} - t_1\mathbf{c}_1)\cdot\mathbf{a}_3^1\big)^2} \\ &= \sqrt{(w_1\mathbf{x} - t_1\mathbf{c}_1)^\top\big(\mathbf{a}_1^1\mathbf{a}_1^{1\top} + \mathbf{a}_2^1\mathbf{a}_2^{1\top} + \mathbf{a}_3^1\mathbf{a}_3^{1\top}\big)(w_1\mathbf{x} - t_1\mathbf{c}_1)} \\ &\le \max\{\|w_1\mathbf{x} - t_1\mathbf{c}_1\|\}\cdot\rho^{1/2}\left(\sum_{i=1}^{3}\mathbf{a}_i^1\mathbf{a}_i^{1\top}\right).\end{aligned} \tag{14}$$

Here, $\rho(\cdot)$ returns the spectral radius of the input matrix. $\sum \mathbf{a}_i^1\mathbf{a}_i^{1\top}$ is clearly symmetric positive definite, so it has positive eigenvalues. Similar to Eq. (12), $\max\{\|w_1\mathbf{x} - t_1\mathbf{c}_1\|\}$ is pre-computed.

The derivation in Eq. (14) can be readily used for bounding homogenous displacement components:

$$\left\|\big(w_1\widetilde{\mathbf{U}}(\mathbf{x}) - t_1\widetilde{\mathbf{U}}(\mathbf{c}_1)\big)\widetilde{\mathbf{q}}^1\right\| \le \max\{\|w_1\widetilde{\mathbf{x}} - t_1\widetilde{\mathbf{c}}_1\|\}\cdot\rho^{1/2}\left(\sum_{i=1}^{3}\widetilde{\mathbf{q}}_i^1\widetilde{\mathbf{q}}_i^{1\top}\right), \tag{15}$$

and heterogenous displacement component:

$$\left\|\big(w_1\widehat{\mathbf{U}}(\mathbf{x}) - t_1\widehat{\mathbf{U}}(\mathbf{c}_1)\big)\widehat{\mathbf{q}}^1\right\| \le \max\{\|w_1\widehat{\mathbf{x}} - t_1\widehat{\mathbf{c}}_1\|\}\cdot\rho^{1/2}\left(\sum_{i=1}^{3}\widehat{\mathbf{q}}_i^1\widehat{\mathbf{q}}_i^{1\top}\right). \tag{16}$$

An upper bound for $\|\Delta\mathbf{u}_2\|$ can be obtained in the same way as in Eqs. (12) to (16). If the vertex is inside a medial slab, we also need to compute the upper bound of $\|\Delta\mathbf{u}_3\|$ for the third handle. Afterwards, we substitute the upper bound of each displacement component of both $\|\Delta\mathbf{u}_1\|$ and $\|\Delta\mathbf{u}_2\|$ into Eq. (10) and update the radius of medial spheres accordingly. In general, a vertex within the MP also receives influential deformation from other out-primitive handles, say $\mathcal{H}_3$. In this case, the vertex displacement becomes: $\mathbf{u} = w_1T_1(\mathbf{x}) + w_2T_2(\mathbf{x}) + w_3T_3(\mathbf{x})$, and we follow the same procedure to derive the upper bound of $w_3T_3(\mathbf{x})$. Should a medial vertex be shared by multiple MPs, the scaling of a medial sphere is determined by the biggest scaling factor among all of its incident MPs.

So far, all the bounding distances are computed based the displacement i.e. how far is the vertex away from its rest-shape position, and we name this MAT update strategy *displacement bounding*. However, it is noticed that displacement bounding tends to yield an excessively scaled MAT during the simulation (i.e. see Fig. 7), which significantly impairs the efficiency of follow-up CC/SCC. Next, we show a tighter bounding method, i.e. deformation bounding, by updating medial spheres within a local frame embedded at each MP.

Deformation bounding If all the vertices within an MP are displaced rigidly, for instance by a certain translation, we do not need to update radii of medial spheres. Nevertheless, because the translation displacement component in Eq. (12) is non-zero, it will scale

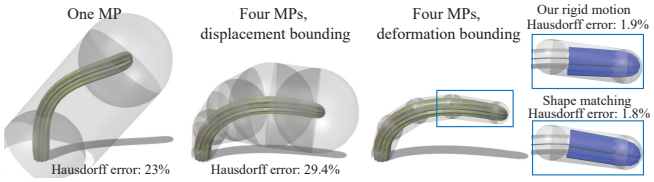

Fig. 7. Displacement bounding scales medial spheres based on vertices' displacement of the MP. As long as the model deviates far from its rest shape, displacement bounding produces a loose enclosure even with little deformation. Deformation bounding fixes this issue by computing a local generalized coordinate with rigid drift factored out.

up the bounding MAT regardlessly. Similarly, a rigid rotation also yields a non-zero affine displacement component and increase the bounding volume. A concrete example is given in Fig. 7, where we bend a cactus stem with four MPs. The top medial cone undergoes a large displacement, and it is rotated for almost 90°. However, compared with the rest-shape geometry, its deformation is rather small. This observation suggests that we could have a much tighter MAT enclosure if the scaling factor is computed based the magnitude of the local deformation other than its global displacement. This is the primary rationale behind our deformation bounding strategy.

The optimal rigid body motion that matches a deformed pose is the one used in shape matching [Müller et al. 2005], which however takes $\mathbf{O}(N)$ to compute the corresponding rotation and translation. Instead, we give up the optimality but compute a good rigid body motion at a lower cost of $\mathbf{O}(n)$, by only using the reduced coordinates at medial handles. The rigid translation $\Delta\bar{\mathbf{c}}$ is the displacement of the geometry center of all the medial spheres on the MP ($\bar{\mathbf{c}}$), and the rigid rotation $\bar{\mathbf{R}}$ is computed by averaging each handle's rotation using Slerp [Shoemake 1985], which can be obtained via the polar decomposition of the handle's affine transformation matrix $\mathbf{A}^j$. Hence, the deformation of a vertex can be formulated as:

$$\mathbf{d} = \mathbf{x} + \sum w_j T^j(\mathbf{x}) - \left[\bar{\mathbf{R}}(\mathbf{x} - \bar{\mathbf{c}}) + \bar{\mathbf{c}} + \Delta\bar{\mathbf{c}}\right]. \tag{17}$$

As quadratic deformation components are less dependent on the underlying rigid body motion, we stick with Eqs. (15) and (16) for computing upper bounds of quadratic DOFs. Therefore, T^j in Eq. (17) only contains translation and affine displacement components at each handle.

The so-called *deformation bounding* computes upper bounds of translation and affine components in a local frame at the center of medial spheres on the MP (i.e. $\bar{\mathbf{c}}$) in order to factor out the rigid body motion hidden in the global displacement. This local frame is also skewed by the corresponding rigid rotation $\bar{\mathbf{R}}$. We use $(\cdot)^*$ to denote a local variable so that $\mathbf{x} = \bar{\mathbf{R}}\mathbf{x}^* + \bar{\mathbf{c}}$, and the local deformation is:

$$\begin{aligned}\mathbf{d}^* &= \bar{\mathbf{R}}^\top\mathbf{d} = \bar{\mathbf{R}}^\top\left(\mathbf{x} + \textstyle\sum w_jT^j(\mathbf{x}) - \bar{\mathbf{R}}(\mathbf{x} - \bar{\mathbf{c}}) - \bar{\mathbf{c}} - \Delta\bar{\mathbf{c}}\right) \\ &= \bar{\mathbf{R}}^\top\Big(\textstyle\sum w_j\mathbf{A}^j + \mathbf{I}\Big)\mathbf{x} + \bar{\mathbf{R}}^\top\textstyle\sum w_j\mathbf{t}^j - (\mathbf{x} - \bar{\mathbf{c}}) - \bar{\mathbf{R}}^\top(\bar{\mathbf{c}} + \Delta\bar{\mathbf{c}}) \\ &= \underbrace{\Big(\bar{\mathbf{R}}^\top\textstyle\sum w_j\mathbf{A}^j\bar{\mathbf{R}} + \mathbf{I} - \bar{\mathbf{R}}\Big)}_{\mathbf{A}^*}\mathbf{x}^* + \underbrace{\bar{\mathbf{R}}^\top\textstyle\sum w_j\big(\mathbf{A}^j\bar{\mathbf{c}} + \mathbf{t}^j\big) - \bar{\mathbf{R}}^\top\Delta\bar{\mathbf{c}}}_{\mathbf{t}^*}.\end{aligned} \tag{18}$$

According to Eq. (18), we obtain the local affine coordinate by concatenating row vectors in $\bar{\mathbf{R}}^\top\sum w_j\mathbf{A}^j\bar{\mathbf{R}} + \mathbf{I} - \bar{\mathbf{R}}$, and $\bar{\mathbf{R}}^\top\sum w_j\big(\mathbf{A}^j\bar{\mathbf{c}} +$

$\mathbf{t}^j) - \overline{\mathbf{R}}^\top \Delta\bar{\mathbf{c}}$ is the local translation coordinate. They are then plugged into Eqs. (12) and (14) to update MPs. Note that both $\max\{|w_j - t_j|\}$ and $\max\{w_j\mathbf{x} - t_j\mathbf{c}_j\}$ are invariant under rigid body motions. Hence, they can still be pre-computed in the global frame.

The bounding quality of the MAT depends on its capability of shape approximation. If we only have few MPs, linearly interpolated spheres may be quite inconsistent with handles' nonlinear deformation, and MAT could be scaled considerably in order to encapsulate the deformed surface. An extreme case is reported in the leftmost sub-figure of Fig. 7, where there is only one medial cone on the MM. The deformation bounding performs no better than the displacement bounding: both have an average Hausdorff error of 23%, due to the limited approximation freedom of MM.

Fortunately, this is seldom the case in practical simulation applications as we always have sufficient MPs to capture desired deformation effects. Under this circumstance, effectively removing the rigid body motion in the displacement plays an important role in tightening the bounding enclosure. As shown in Fig. 7, even with extra three MPs added, naïve displacement bounding still has a higher Hausdorff error of 29.4%. On the other hand, our deformation bounding strategy provides a high-quality MAT bounding, which has the Hausdorff error of 1.9%. It is only 0.1% higher than the Hausdorff error based on the shape matching, which is typically considered optimal. Yet, our deformation bounding is much more efficient and updates MAT at $\mathbf{O}(n)$ since all the computations (i.e. see Eq. (18)) only need the generalized coordinate. Another example of a pendent Armadillo is given in Fig. 8, which is consistent with the cactus example. Deformation bounding tightly encapsulates the deformed Armadillo with a very small Hausdorff error (2.1%). The remaining question is: *how can we efficiently detect intersections between MPs?* We elaborate on this important technical challenge in the next section.

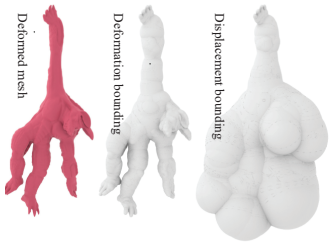

Fig. 8. With 104 handles, deformation bounding well encapsulates the deformed Armadillo. The average Hausdorff error is only 2.1%. The naïve displacement bounding produces a very loose enclosure with the Hausdorff error of 18.2%

5 COLLISION CULLING AND DETECTION

In order to perform CC/SCC effectively, we need to know if two MPs overlap with each other (for collision culling), and if so, where is the deepest intersection so that we can directly identify nearby triangles and facilitate the follow-up collision processing. Each intersecting MP can be either a medial cone or a medial slab (i.e. Fig. 3). We know that both types of MPs are surfaces of interpolated spheres, which can be naturally expressed using implicit surface functions. Computing the minimum distance between two implicit surfaces is a difficult geometric problem [Chen et al. 2006]. In our case, as MP surface is quadratic, it can be formulated as a quadratically constrained quadratic program (QCQP) problem. Thanks to simplex-interpolated MPs, the number of unknowns remains manageable, and we show that this problem can be exactly solved.

5.1 Intersection Test between Medial Cones

Consider two medial cones C_1 and C_2. Let $\mathbf{c}_{1,1}$ and $\mathbf{c}_{1,2}$ be the centers of medial spheres at two medial vertices of C_1, whose radii are $r_{1,1}$ and $r_{1,2}$ respectively. Similarly, $\mathbf{c}_{2,1}$, $\mathbf{c}_{2,2}$ and $r_{2,1}$, $r_{2,2}$ are sphere centers and radii of C_2. $0 \le t_1 \le 1$ is the interpolation parameter of C_1, and $0 \le t_2 \le 1$ is the interpolation parameter of C_2. In C_1, each point along the line segment connecting $\mathbf{c}_{1,1}$ and $\mathbf{c}_{1,2}$ defines a sphere center (and so is in C_2) such that:

$$\mathbf{c}_1 = t_1\mathbf{c}_{1,1} + (1-t_1)\mathbf{c}_{1,2}, \text{ and } \mathbf{c}_2 = t_2\mathbf{c}_{2,1} + (1-t_2)\mathbf{c}_{2,2}. \quad (19)$$

Their radii are also linearly interpolated:

$$r_1 = t_1 r_{1,1} + (1-t_1)r_{1,2}, \text{ and } r_2 = t_2 r_{2,1} + (1-t_2)r_{2,2}. \quad (20)$$

The minimum surface distance between C_1 and C_2 can be formulated as the following QCQP:

$$\begin{aligned} \min \quad & d(t_1,t_2) = \|\mathbf{x}_1 - \mathbf{x}_2\| \\ \text{s.t.} \quad & (\mathbf{x}_1 - \mathbf{c}_1)\cdot(\mathbf{x}_1 - \mathbf{c}_1) = r_1^2, \quad (\mathbf{x}_2 - \mathbf{c}_2)\cdot(\mathbf{x}_2 - \mathbf{c}_2) = r_2^2, \\ & 0 \le t_1 \le 1, \quad 0 \le t_2 \le 1. \end{aligned} \quad (21)$$

Here $\mathbf{x}_1$ and $\mathbf{x}_2$ represent two arbitrary points on the surface of C_1 and C_2. Both of them are quadratically constrained by interpolated sphere surfaces. $\mathbf{c}$ and r are functions of the parameters t_1 and t_2, which are box-constrained between 0 and 1. Due to the geometric symmetry of an MP, we can avoid referring to Lagrange multipliers and simplify the solving process. The *signed* surface distance between C_1 and C_2 can always be written as the distance between $\mathbf{c}_1(t_1)$ and $\mathbf{c}_2(t_2)$ (denoted with $\sqrt{S}$) minus the sum of corresponding medial radii. Therefore, the QCQP of Eq. (21) can be reduced to the following minimization problem:

$$\begin{aligned} \min \quad & f(t_1,t_2) = \|\mathbf{c}_1 - \mathbf{c}_2\| - (r_1 + r_2) = \sqrt{S} - (R_1t_1 + R_2t_2 + R_3) \\ \text{s.t.} \quad & 0 \le t_1 \le 1, \quad 0 \le t_2 \le 1, \end{aligned} \quad (22)$$

where

$$\begin{aligned} & S = At_1^2 + Bt_1t_2 + Ct_2^2 + Dt_1 + Et_2 + F, \\ & A = (\mathbf{c}_{1,1} - \mathbf{c}_{1,2})\cdot(\mathbf{c}_{1,1} - \mathbf{c}_{1,2}), \quad B = -2(\mathbf{c}_{1,1} - \mathbf{c}_{1,2})\cdot(\mathbf{c}_{2,1} - \mathbf{c}_{2,2}), \\ & C = 2(\mathbf{c}_{2,1} - \mathbf{c}_{2,2})\cdot(\mathbf{c}_{2,1} - \mathbf{c}_{2,2}), \quad D = 2(\mathbf{c}_{1,1} - \mathbf{c}_{1,2})\cdot(\mathbf{c}_{1,2} - \mathbf{c}_{2,2}), \\ & E = -2(\mathbf{c}_{2,1} - \mathbf{c}_{2,2})\cdot(\mathbf{c}_{1,2} - \mathbf{c}_{2,2}), \quad F = (\mathbf{c}_{1,2} - \mathbf{c}_{2,2})\cdot(\mathbf{c}_{1,2} - \mathbf{c}_{2,2}), \\ & R_1 = r_{1,1} - r_{1,2}, \quad R_2 = r_{2,1} - r_{2,2}, \quad R_3 = r_{1,2} + r_{2,2}. \end{aligned}$$

Cone-cone collision culling Unless $f(t_1,t_2)$ in Eq. (22) is less than or equal to zero, from which we know C_1 and C_2 do not overlap, and their actual nearest distance is not of our interest. This is checked by quickly identifying if there exists at least one real solution for $f(t_1,t_2) = 0$, in order to secure a collision. Because both $\|\mathbf{c}_1 - \mathbf{c}_2\|$

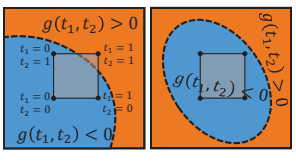

Fig. 9. Two medial cones intersect when $g(t_1,t_2) = 0$ overlaps with the constraint region (left), or one medial cone is completely inside of the other (right).

and $r_1 + r_2$ are positive, $f(t_1, t_2) = 0$ is equivalent to:

$$g(t_1, t_2) = \|\mathbf{c}_1 - \mathbf{c}_2\|^2 - (r_1 + r_2)^2 = A't_1^2 + B't_1t_2 + C't_2^2 + D't_1 + E't_2 + F' = 0. \quad (23)$$

$g(t_1, t_2)$ is essentially a bivariate quadratic function of t_1 and t_2 with its 6 coefficients defined as: $A' = A - R_1^2$, $B' = B - R_1R_2$, $C' = C - R_2^2$, $D' = D - R_1R_3$, $E' = E - R_2R_3$, $F' = F - R_3^2$. Setting g to zero describes a quadratic curve of the quadric-plane intersection, known as the *conic section*. The conic section is either an ellipse, a parabola, or a hyperbola, and it divides the 2D parametric domain of t_1 and t_2 into positive and negative regions as shown in Fig. 9. $g(t_1, t_2) = 0$ has a solution if and only if the conic section overlaps with the boxed region of $t_1, t_2 \in [0, 1]$. Alg. 1 describes how it can be efficiently processed. Specifically, we start with testing $g(t_1, t_2)$ at four corners of the boxed constraint: $g(0, 0)$, $g(0, 1)$, $g(1, 0)$, $g(1, 1)$. If any of them is less than or equal to zero, we know C_1 and C_2 intersect with each other (lines 1–3 in Alg. 1). Afterwards, $g(t_1, t_2) = 0$ is tested for each edge of the constraint area (i.e. $t_1 = 0$ or 1, and $t_2 = 0$ or 1). This is done by solving a quadratic equation of t_1 (or t_2) with t_2 (or t_1) setting as 0 or 1 respectively (lines 8–19 in Alg. 1). Note that there exists a special case that is not yet covered. This is when $g(t_1, t_2) = 0$ forms an ellipse which encloses the constraint region, or the constraint box encloses the ellipse (i.e. see Fig. 9 right). Geometrically, this corresponds to the situation where a medial cone sits inside of the other cone completely. Therefore, we need to verify that the interior of the ellipse is negative, and the center of the ellipse is within $t_1, t_2 \in [0, 1]$ to secure a collision (lines 20–22 in Alg. 1). If Alg. 1 returns a false, we know C_1 and C_2 do not collide with each other and the corresponding collision is culled.

This routine can be readily generalized for CC between other MPs like cone-slab CC and slab-slab CC. In these cases, $g = 0$ cannot be intuitively visualized as a 2D conic section. However, the problem structure is unchanged, and we can project this higher-dimension problem to a lower-dimension space. Specifically, we still start with corner points of the boxed constraint (which are 3D or 4D points). If any negative g is observed, a collision is recognized. After that, we test the intersection between $g = 0$ and edges, planes, or hyperplanes of the constraint region (i.e. by setting the corresponding parameter to 0 or 1). This eventually brings us a quadratic equation to solve. As long as it has one root between 0 and 1, the collision is recognized. We also need to check the extreme situation when the 3D ellipsoid or 4D hyperellipsoid encloses the constraint region. While it seems obscure, we just test g value at the center of the ellipsoid or hyperellipsoid. If it is negative, we further verify if it is inside the constraint region to secure a collision – just as we do for the cone-cone CC.

Cone-cone collision detection After Alg. 1 confirms a collision between C_1 and C_2, our CD algorithm efficiently computes their deepest penetration. We start by checking if $4AC - B^2 = 0$ as our **case 0** (i.e. see Fig. 10). If $4AC - B^2 = 0$, $\mathbf{c}_{1,2} - \mathbf{c}_{1,1}$ and $\mathbf{c}_{2,2} - \mathbf{c}_{2,1}$ are parallel to each other. In this case, S does not contribute to the minimization of $f(t_1, t_2)$ because either t_1 or t_2 is able to minimize S at its entire span of $[0, 1]$. In other words, $\partial S/\partial t_1$ and $\partial S/\partial t_2$ become linearly dependent to each other leading to infinitely many

ALGORITHM 1: Cone-cone collision culling.

Input: $A' = A - R_1^2$, $B' = B - R_1R_2$, $C' = C - R_3^2$, $D' = D - R_1R_3$, $E' = E - R_2R_3$, $F' = F - R_3^2$ for C_1 and C_2
Output: if C_1 and C_2 collide with each other or not

```
1:  if g(0,0) ≤ 0 or g(0,1) ≤ 0 or g(1,0) ≤ 0 or g(1,1) ≤ 0 then
2:  |  return True
3:  end
4:  solve A't1^2 + D't1 + F' = 0; // set t2 = 0
5:  if t1 ∈ [0,1] then
6:  |  return True
7:  end
8:  solve A't1^2 + (B'+D')t1 + C' + E' + F' = 0; // set t2 = 1
9:  if t1 ∈ [0,1] then
10: |  return True
11: end
12: solve C't2^2 + E't1 + F' = 0; // set t1 = 0
13: if t2 ∈ [0,1] then
14: |  return True
15: end
16: solve C't2^2 + (B'+E')t2 + A' + D' + F' = 0; // set t1 = 1
17: if t2 ∈ [0,1] then
18: |  return True
19: end
    /* g(t1,t2) = 0 must be an ellipse                          */
20: (tx, ty) ← center of g(t1,t2) = 0;
21: if tx, ty ∈ [0,1] and g(tx,ty) < 0 then
22: |  return True
23: end
24: return False
```

minimizers of S. Thus, r_1 and r_2 fully determine the minimum value of $f(t_1, t_2)$, and the global minimizer of $f(t_1, t_2)$ must be at one of the four medial vertices on C_1 and C_2. Otherwise $4AC - B^2 > 0$, and we follow the standard routine of setting the first-order partial derivatives of f w.r.t. the unknown parameters t_1 and t_2 to zeros:

$$\begin{cases} \dfrac{\partial f(t_1, t_2)}{\partial t_1} = \dfrac{1}{2\sqrt{S}} \cdot (2At_1 + Bt_2 + D) - R_1 = 0, \\ \dfrac{\partial f(t_1, t_2)}{\partial t_2} = \dfrac{1}{2\sqrt{S}} \cdot (Bt_1 + 2Ct_2 + E) - R_2 = 0. \end{cases} \quad (24)$$

Due to the existence of $1/\sqrt{S}$, Eq. (24) has a singular point at (t_1^*, t_2^*). If this happens, and $t_1^*, t_2^* \in [0, 1]$ satisfy the box constraint, (t_1^*, t_2^*) is the minimizer of f. In all other situations, we solve Eq. (24) by checking the radius's variation of C_1 and C_2 as shown in Fig. 10. In most cases, we end up with solving a quadratic equation of either t_1 or t_2 in the form of:

$$X_1t_1^2 + X_2t_1 + X_3 = 0, \quad \text{or} \quad Y_1t_2^2 + Y_2t_2 + Y_3 = 0. \quad (25)$$

As a quadratic equation has two possible roots in general, we check both possibilities and pick the one yielding a smaller f value.

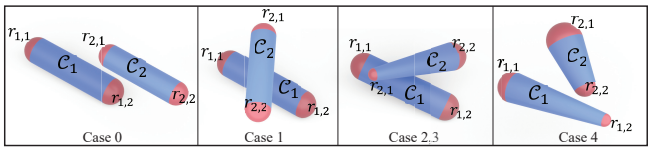

Fig. 10. We handle the cone-cone intersection test by checking radius variations at C_1 and C_2. Each case leads to a closed-form formulation of the signed minimum distance between C_1 and C_2.

Case 1 $R_1 = 0$ and $R_2 = 0$. $R_1 = 0$ and $R_2 = 0$ suggest the medial radii of both cones are constant. As a result, Eq. (24) becomes a linear system, and it gives $t_1 = BE - 2CD/4AC - B^2$ and $t_2 = BD - 2AE/4AC - B^2$.

Case 2 $R_1 = 0$ and $R_2 \neq 0$. $R_1 = 0$ means C_1 does not have radius variation, and $\partial f(t_1, t_2)/\partial t_1 = 0$ becomes a linear equation of $t_1 = H_1 t_2 + K_1$, where $H_1 = -B/2A$, $K_1 = -D/2A$. We solve t_2 using Eq. (25) where:

$$
\begin{aligned}
Y_1 &= (2C + BH_1)^2 - 4R_2^2(AH_1^2 + BH_1 + C),\\
Y_2 &= 2(2C + BH_1)(BK_1 + E) - 4R_2^2(2AH_1K_1 + BK_1 + DH_1 + E),\\
Y_3 &= (BK_1 + E)^2 - 4R_2^2(AK_1^2 + DK_1 + F).
\end{aligned}
$$

Case 3 $R_1 \neq 0$ and $R_2 = 0$. Case 3 is symmetric to case 2, where we have $t_2 = H_2 t_1 + K_2$ for $H_2 = -B/2C$ and $K_2 = -E/2C$. t_1 is computed via Eq. (25), and the coefficients are:

$$
\begin{aligned}
X_1 &= (2A + BH_2)^2 - 4R_1^2(A + BH_2 + CH_2^2),\\
X_2 &= 2(2A + BH_2)(BK_2 + D) - 4R_1^2(BK_2 + 2CH_2K_2 + D + EH_2),\\
X_3 &= (BK_2 + D)^2 - 4R_1^2(CK_2^2 + EK_2 + F).
\end{aligned}
$$

Case 4 $R_1 \neq 0$ and $R_2 \neq 0$. This is the most general situation for the cone-cone test. When $R_1 \neq 0$ and $R_2 \neq 0$, Eq. (24) can be manipulated to yield:

$$
\begin{aligned}
&\frac{2At_1 + Bt_2 + D}{Bt_1 + 2Ct_2 + E} = \frac{2R_1\sqrt{S_c}}{2R_2\sqrt{S_c}} = \frac{R_1}{R_2}\\
\Rightarrow\ &(2At_1 + Bt_2 + D)R_2 = (Bt_1 + 2Ct_2 + E)R_1\\
\Rightarrow\ &(2AR_2 - BR_1)t_1 = (2CR_1 - BR_2)t_2 + (ER_1 - DR_2),
\end{aligned}
$$

or simply: $L_1 t_1 = L_2 t_2 + L_3$. If $L_1 = 0$ and $L_2 = 0$, it can be verified that $4AC - B^2 = 0$, and it becomes **case 0**. If $L_1 = 0$, $L_2 \neq 0$ or $L_1 \neq 0$, $L_2 = 0$, we can directly obtain $t_2 = -L_3/L_2$ or $t_1 = L_3/L_1$. Otherwise, if $L_1 \neq 0$ and $L_2 \neq 0$, we have $t_1 = L_2/L_1 t_2 + L_3/L_1 = H_3 t_2 + K_3$. Thus, t_2 can be computed via Eq (25), where all the coefficients are:

$$
\begin{aligned}
Y_1 &= (2C + BH_3)^2 - 4R_2^2(AH_3^2 + BH_3 + C),\\
Y_2 &= 2(2C + BH_3)(BK_3 + E) - 4R_2^2(2AH_3K_3 + BK_3 + DH_3 + E),\\
Y_3 &= (BK_3 + E)^2 - 4R_2^2(AK_3^2 + DK_3 + F).
\end{aligned}
$$

5.2 Collision Detection between Other Medial Primitives

For the cone-cone test, each cone has either a constant or linear radius variation, which leads to 4 different cases. A slab has two independent edges, which leads to 4 radius variation patterns. In theory, we will have $2\times4 = 8$ cases for a cone-slab test and $4\times4 = 16$ cases for a slab-slab test. However, one may notice that, many of them are *symmetric* sharing identical problem structure, just as **case 2** and **case 3** in the cone-cone test. We now show how to identify those potentially redundant cases, and generalize to an efficient intersection test to other types of MPs.

For an independent medial edge $\mathcal{E}_k$ on the primitive with the interpolation parameter t_k, we use a binary digit B_k to denote its radius variation status: $B_k = 0$ means we have a constant radius size along $\mathcal{E}_k$, and $B_k = 1$ suggests the radius linearly varies. Note that $B_k = 0$ also implies the corresponding partial derivative $\partial f/\partial t_k$ degenerates to a linear constraint, while $B_k = 1$ implies $\partial f/\partial t_k$ remains a quadratic one. The fact is that we use the same computational routine to solve all the t_k as long as the total numbers of quadratic equations and linear equations are the same, regardless of their indexing order. For example, the binary encodings for **case 1** and **case 4** in the cone-cone test (where we have two edges in total) are 00 and 11. In the meantime, the binary encodings for **case 2** and **case 3** are 01 and 10. Therefore, **case 2** and **case 3** are symmetric and can be handled collectively. For a cone-slab test, because there are 3 independent edges we need 4 cases to fully process the CD. They are binarily coded as 000, 001, 011, and 111, corresponding to systems with 0, 1, 2 and 3 quadratic constraints. Similarly, there will be 5 different cases for a slab-slab test, represented as 0000, 0001, 0011, 0111 and 1111.

Solving a system of quadratic equations is handled by converting it to a linear system except that each quadratic constraint potentially leads to two different linear constraints. For instance, in **case 2** and **case 3** of the cone-cone test, each root of the quadratic equation imposes a linear constraint between t_1 and t_2, which can be finally solved by coupling with the remaining linear constraints. Therefore, in the worse case where we have all $B_k = 1$, we could have 2^K possible solutions, where $K = \{2, 3, 4\}$ denotes the total number of independent edges in the test. Fortunately, MAT only uses 2D simplex interpolations. The total number of DOFs involved in a test is at most 4 (the slab-slab test). Therefore, it can still be efficiently solved on GPU.

5.3 Implementation Details

We couple MAT-based enveloping with spatial subdivision [Pabst et al. 2010], that voxelizes the entire animation scene. After Alg. 1 recognizes a collision, our CD algorithm instantly gives the coordinate of the deepest intersection point. This location is mapped to the voxel index, and we query for all the surface triangles that overlap with this voxel. The size of each voxel is set as the length of the longest triangle edges on the model's surface. As a result, this query only returns very few triangles. Note that the actual voxel grid is never created, and what we need is just an `unsigned int` triple that represents the voxel index along three dimensions. The triangle-voxel overlapping test is simplified as the overlapping test between voxel and the spatial span of the triangle in all three directions, which can be performed using only `boolean` operations. The standard triangle-triangle collision test is then followed.

We pre-build a list of collision test including all MP pairs between two models for CC as well as all the MP pairs on a model itself for SCC. Because the medial sphere on a medial vertex is shared by multiple MPs, it will release fake collision signals if not properly dealt with. To this end, we skip all the possible fake collisions from the shared medial sphere when constructing our list of collision test by assuming as all the triangles within a sphere do not collide. The total time complexity of our MAT-based CC/SCC is $\mathbf{O}(k^2n^2)$ where k denotes the number of objects in the scene. All the MP collision tests are performed on the GPU in parallel.

6 EXPERIMENTAL RESULTS

We implement our framework on a `Windows` desktop computer with an `Intel i7 5960X` CPU (3.0 GHz) and an `nVidia Titan RTX` GPU

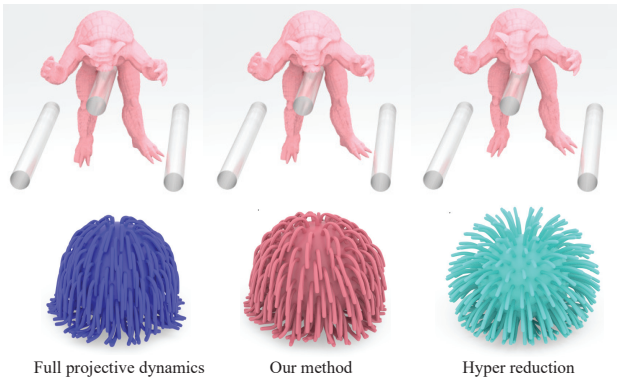

Fig. 11. The Armadillo model has smoother geometry than the puffer ball, and hyper-reduced solver yields a more natural result (top). Likewise, reducing the material stiffness also relieves the stiffening artifact induced by the local reduction (bottom). However, the difference between full/semi-reduced solver and hyper-reduced simulation is still perceivable.

(16 GB memory). We tested our system on a variety of geometrically complex models in collision-rich scenes. Our system produces high-quality animations with large deformation and well-preserved local details at an interactive rate, including the collision handling. We refer the reader to the supplementary materials for more details, which include **video, executables, and source code**.

Model preparation The initial Voronoi tessellation for MM construction is generated using the CGAL library [Fabri and Teillaud 2011]. The biharmonic weight is then computed based on the simplified MAT [Jacobson et al. 2011]. Tab. 1 reports detailed timing statistics at this stage as well as models' geometry information. Both Voronoi tessellation and biharmonic weight computations are on the CPU. Here, the Voronoi tessellation is the most time-consuming procedure, which takes up to tens of minutes. The weight calculation is also quite expensive, but it can be processed in parallel with CPU multi-threading. After MMs are ready, our system advances to the online simulation stage, where all the computations are carried out on the GPU with nVidia cuBLAS.

Animation quality Next, we take a closer look over the animation quality of our semi-reduced projective dynamics solver. In terms of the simulation algorithm, our most relevant competitor is the hyper-reduced solver [Brandt et al. 2018]. To further validate our method, we simulate a falling Armadillo (with 1,000 constraint samples for the hyper-reduced solver), which interacts with three glassy rods. As we can see from Fig. 11 (top), the visual difference between our method and the hyper-reduced solver is smaller than the puffer ball example in Fig. 5. This is because the local reduction of the hyper-reduced solver induces

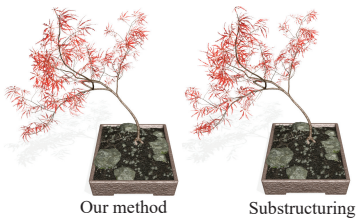

Fig. 12. Our method produces plausible animations of the maple bonsai when a sinusoidal wind is applied. This result is comparable to the substructuring method [Barbič and Zhao 2011].

	# Ele.	# Tri.	# MH	MA	QM	BW
Puffer ball	$610k$	$120k$	1,257|12	776	73	13
Ship	$2,526k$	$236k$	188|9	640	64	4
Staypuft	$462k$	$236k$	21|7	172	10	< 1
Armadillo	$156k$	$62k$	104|26	362	6	< 1
Dinosaur	$194k$	$114k$	67|20	470	31	2
Cactus (1)	$231k$	$139k$	208|22	216	23	2
Cactus (2)	$243k$	$115k$	183|18	234	23	1
Cactus (3)	$95k$	$58k$	19|2	186	9	< 1
Cactus (4)	$34k$	$13k$	26|2	64	3	< 1
Cactus (5)	$47k$	$24k$	35|4	86	3	< 1
Cactus (6)	$37k$	$12k$	23|2	52	2	< 1
Cactus (7)	$26k$	$6k$	21|2	36	2	< 1
Cactus (8)	$13k$	$3k$	7|2	24	1	< 1

Table 1. Time statistics of constructing MMs and computing the weight coefficients. # **Ele.** and # **Tri.** are total numbers of tetrahedra on the simulation mesh and triangle faces on the surface. # **MH** records the total number of medial handles placed (# affine MHs | # quadratic MHs). **MA** stands for the time for computing the initial MA and the Voronoi tessellation. **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.

lower residual errors in this example, due to smoother and less concave geometry of the Armadillo (the residual error is still twice bigger than our method though). Similarly, softening the material also improves the plausibility of the hyper-reduced simulation. After we make the puffer ball three times softer (Fig. 11, bottom), the stiffening artifact of the hyper-reduced solver becomes less severe compared with Fig. 5 as tuning down the stiffness also reduces the residual error for the local reduction. Yet, our method still gives better results that are closer to the full simulation in both examples.

As a spatial reduction simulator, our method is also similar to multi-domain simulation [Barbič and Zhao 2011; Yang et al. 2013] as one may consider each MH a sub-domain on the deformable object, and the handle's DOFs prescribe local subspace deformations. To this end, we compare our method with the deformation substructuring method [Barbič and Zhao 2011]. The snapshots are provided in Fig. 12, where the maple bonsai model is decomposed into 1,771 sub-domains using the substructuring method and 1,771 handles (i.e. one handle per domain) using our method. Both simulations yield natural animations when a sinusoidal wind field is applied. In this example, self-collision is not processed. While MAT-based model reduction produces satisfying animation in general, it could be cumbersome when there exist many compression effects in the animation. As reported in Fig. 13, we can see some unnatural shapes at the bottom

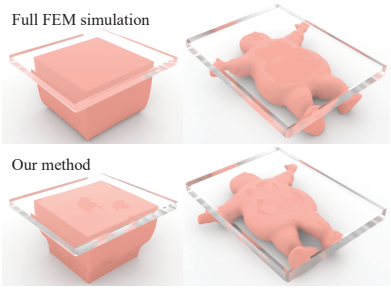

Fig. 13. Artifacts can be seen in the compression due to the lack of deformation DOFs for the bulging effect in MAT-based reduction.

of the box when it is pressed. In this example, only three quadratic MHs are used, and the MAT subspace does not have sufficient freedoms to capture the volumetric bulging during the compression. Adding more handles on the Staypuft model yields a better result.

MAT for other materials Medial elastics synergizes well with the projective dynamics, because both reduced simulation and CC/CD can be effectively accelerated by the GPU. Nevertheless, MAT can also handle other hyperelastic materials following the standard model reduction procedure [Sifakis and Barbic 2012] by projecting Eq. (4) into the column space of $\mathbf{U}$ (i.e. Eq. (2)). Fig. 14 shows snapshots of five falling Armadillos. In this experiment, each Armadillo is assigned with a different material including StVK, Co-rotational, Neo-Hookean, Mooney-Rivlin, and Arruda-Boyce. We also pre-compute Cubature [An et al. 2008] for an efficient run-time estimation of reduced internal force and tangent stiffness matrix. This example is less demanding than other examples as in Figs. 1 and 18, however the FPS is even slower (only 1.2 FPS). This is because the simulation leads to a time-varying dense system, which is not GPU-friendly. As a result, we need to use CPU for the simulation and pass the resulting reduced coordinate to the GPU for MAT-based CC/CD. Frequent communications between the CPU and GPU cause extra overheads.

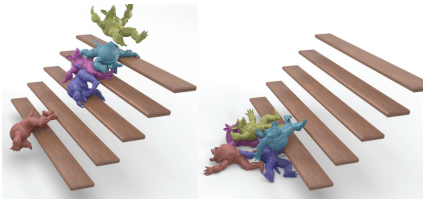

Fig. 14. One can also treat MAT as a general model reduction method. In this experiment, five Armadillos fall on staircases and collide with each other. Each Armadillo is assigned with a different material, in the order of: StVK, Co-rotational, Neo-Hookean, Mooney-Rivlin, and Arruda-Boyce.

	Subspace	Fact.	# Cons.	Global	Local
Puffer ball	15, 444	3.00	610, 452	2.43	0.62
Ship	2, 526	0.67	482, 052	0.38	0.56
Staypuft	462	0.16	212, 608	0.12	0.31
Armadillo	2, 028	0.62	65, 013	0.32	0.16
Dinosaur	1, 404	0.45	193, 695	0.24	0.28
Cactus (1)	5, 076	0.87	231, 312	0.49	0.33
Cactus (2)	4, 812	0.85	243, 552	0.46	0.39
Cactus (3)	288	0.14	94, 740	0.11	0.19
Cactus (4)	372	0.17	34, 164	0.12	0.14
Cactus (5)	540	0.23	46, 584	0.13	0.14
Cactus (6)	336	0.15	37, 458	0.10	0.13
Cactus (7)	312	0.19	26, 412	0.10	0.12
Cactus (8)	240	0.06	12, 942	0.09	0.10

Table 2. Efficiency benchmarks of our semi-reduced projective dynamics simulator. **Subspace** is the size of the MM-based subspace. **Fact.** is the time used for pre-factorizing the reduced global matrix. # **Cons.** is the total number of constraints of the model. **Global** is the time for solving the pre-factorized global matrix. **Local** is the computation time for all the local projections. All the timing units are in ms.

Culling effectiveness Fig. 15 reports a collision-heavy example, where the dinosaur model falls into a group of cacti. The initial dinosaur-cactus collision induces a sequence of follow-up collisions between cacti. The animation is around 18 FPS including the simulation, CC/SCC, and collision resolve. The subspace sizes of the dinosaur and all the eight cacti are reported in Tab. 2. In addition to the animation itself, we also examine the effectiveness among AABBs, bounding spheres and MAT based CC/SCC for this scene. Here, we set up bounding AABBs and spheres using two strategies: 1) AABBs/spheres enclose the model as tightly as the MAT does (i.e. with the same Hausdorff error); or 2) the total numbers of AABBs/spheres are the same as the number of MPs. For a clearer visualization, we only color the leaf AABBs or spheres that participate in the collision. For the MAT-based CC and SCC, we just highlight the voxel, which envelops the deepest interpenetration between two MPs. The deformed MATs of all the cacti are also shown in the figure (the second row).

Detailed collision statistics are given in Fig. 16. When AABBs or spheres achieve the same bounding quality of MAT (fourth and sixth rows in Fig. 15), the BVH is deep, and CC involves orders-of-magnitude more primitive-wise intersection tests along the hierarchy than the MAT. On the other hand, if we set the total number of leaf primitives equivalent to the number of MPs, the BVH is shallower, but each leaf AABB or sphere becomes bigger (i.e. see the third and fifth rows in Fig. 15), which encapsulates much more surface triangles for the CD. While better than the first strategy, it is also less efficient than our method. Thanks to its superior compactness, MAT outperforms conventional BVHs of AABBs or spheres – only triangles that overlap the voxel of deepest interpenetration need to be tested for CD. Moreover, it is reasonable that self-collision of the triangles within an MP could be neglected because of as-rigid-as possible constraints are forced in the system.

To objectively quantify the collision performance, we compare the *culling effectiveness* for different bounding primitives, which is defined as the ratio between the total number of intersected primitives and the total number of intersected triangles during CC. The result can be found in the top-right plot of Fig. 16. It is clear that MAT-based CC/SCC is much more efficient than traditional BVHs. It is also worth mentioning that our method does not have tree-like multi-level data structures. This is adequate in practice because an MAT will have a good shape approximation only with a modest number of medial vertices, and we can fully exploit modern GPUs to efficiently run the overlapping tests for all MP pairs in parallel.

	Global	Local	Sim.	CC/SCC	CD	FPS
Fig. 1	0.38	0.56	42.71	18.7	3.1	15.3
Fig. 15	1.8	1.1	34.5	17.1	1.5	17.7
Fig. 18 (left)	1.9	1.7	39.6	18.3	2.4	15.2
Fig. 18 (right)	14.8	3.7	76.5	21.6	4.8	9.7

Table 3. Time breakdown of examples in Fig. 1, Fig. 15, and Fig. 18. **Global** and **Local** are the time used for the (reduced) global and (full) local steps of one iteration. **Sim.** is the total time of our semi-reduced solver (which includes extra computational overheads e.g. for projecting generalized coordinate to fullspace, etc). **CC/SCC** is the culling time.**CD** is the collision detection time. All the timing units are in ms. The average FPS for the overall animation is in the column **FPS**.

Displacement and deformation bounding We prefer deformation bounding for a much tighter bounding. Displacement bounding

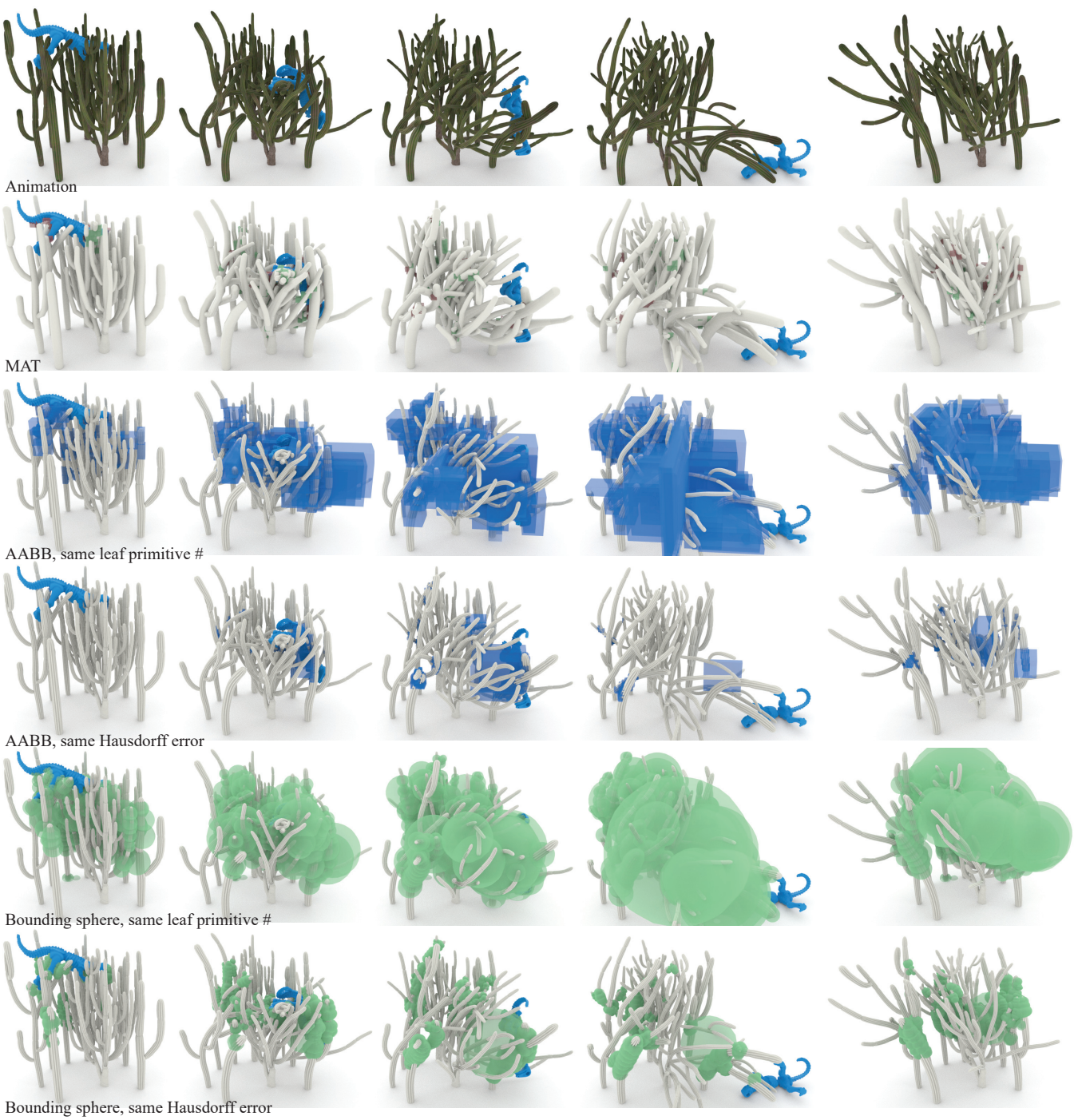

Fig. 15. The dinosaur model falls through a group of eight cacti. We compare our MAT-based CC with BVHs of AABBs and bounding spheres. This animation runs at ~ 18 FPS with all the collisions and self-collisions culled, detected, and resolved. MAT-based CC/SCC is highlighted in the second row, where the deformed MATs of all the cacti are also given. For BVHs of AABBs and spheres, we either set the BVH to have the same bounding quality as MAT (same Hausdorff error) or to have the same number of leaf primitives as MPs on the MAT (same leaf primitive #). Only leaf-level AABBs and spheres are highlighted.

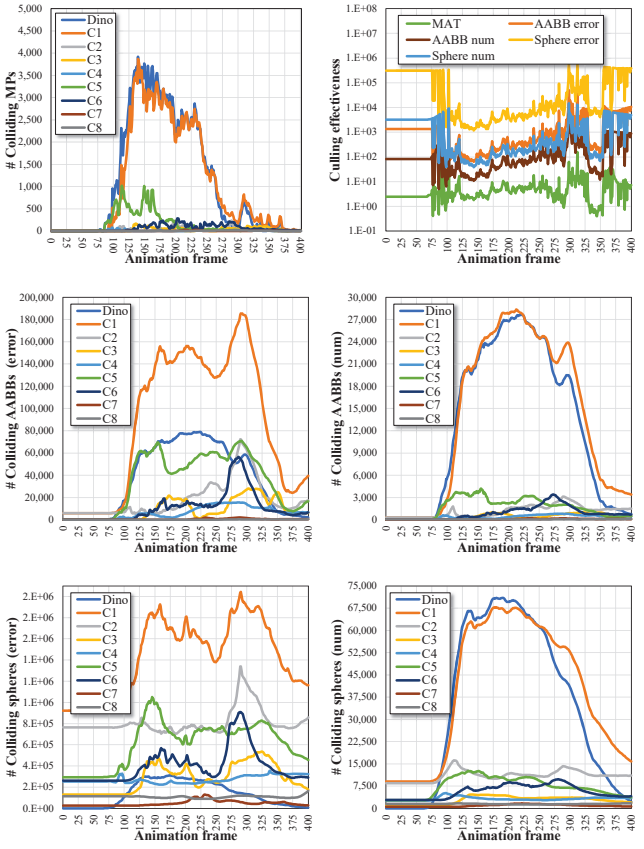

Fig. 16. We plot the detailed benchmarks for BVHs of AABBs and spheres at each individual deformable body in the falling dinosaur animation (Fig. 15). We also compare the culling effectiveness of different bounding primitives in the top-right plot, where **error** is for primitives with the same Hausdorff error, and **num** is for the same numbers of leaf primitives.

yields a loose MAT enclosure as long as there exist large rigid body motions. For highly reduced models with limited MHs, the bounding quality could be low (i.e. see Fig. 17). Fortunately, we seldom want to do so because an extravagant model reduction also impairs the simulation quality. As long as we have a moderate number of MHs, a tight bounding could be always obtained using the deformation bounding. To elaborate it, we plot the average Hausdorff errors of eight cacti in the dragon-cacti collision scene using different numbers of MHs for both displacement and deformation bounding. Obviously, the displacement bounding errors are much higher than deformation bounding.

More examples More examples are shown in Fig. 1 and Fig. 18 with per-step time breakdowns in Tab. 3. In Fig. 1, a barbarian ship model falls over a few glassy rods. This ship has $482k$ elements and $236k$ triangles. Its subspace size is 2, 526 with 482, 052 constraints. As it slides down to the floor, we observe rich contacts and collisions between the ship and rods. Each side of the ship has a row of soft paddles, which collide with each other with interesting local deformations. Our solver captures all of these details, and the animation is $\sim$ 15 FPS. Fig. 18 (left) is an enhanced scene of Fig. 15, where we

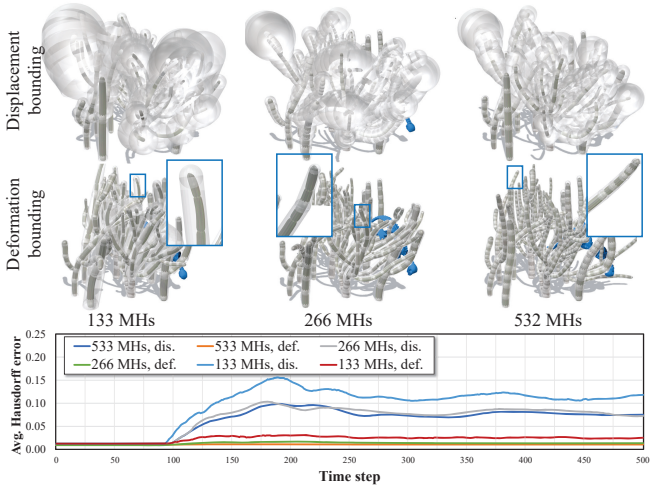

Fig. 17. Even with sparse MHs deployed, deformation bounding still yields high-quality MAT enclosures, while the displacement bounding scales MPs excessively. The average Hausdorff error variation during the animation is also plotted.

have four Staypufts flying into the cactus bush. The cactus spines pierce the Staypuft after hard collisions. The subspace size at each Staypuft is 462, and there are 212, 608 constraints. Cacti swing and interact with each other vibrantly in this example. Our solver runs at $\sim$ 15 FPS.

A more challenging scenario is shown in Fig. 18 (right), where six puffer balls fall one-by-one and collide with each other. Each puffer ball also has many self-collisions among its elastic strings. We consider it a good example to showcase the benefit of MAT. While geometrically complex, each elastic string on a puffer ball can be tightly encapsulated by only few MPs, and CC/SCC can be efficiently handled. In this example, each puffer ball has as many as 610, 452 constraints. Yet, the local step only takes 1.6 *ms* for each ball. On the other hand, the subspace size at each puffer ball reaches 15, 444. Solving such big dense matrix is prohibitive if an interactive rate is desired, even with CUDA. Therefore, we prune our weight functions by explicitly enforcing its locality as in [Brandt et al. 2018] so that each MH only influences its nearby mesh vertices. Doing so makes the reduced global matrix block-sparse so that we can use the sparse Cholesky to process the global step more efficiently. This animation runs at $\sim$ 10 FPS on average. When intensive collisions occur among multiple puffer balls, the FPS drops to 5 due to slower convergency induced by collision constraints. The average Hausdorff errors of these three experiments are plotted in Fig. 19.

7 LIMITATION AND FUTURE WORK

In this paper, we present a framework to simulate nonlinear dynamics of deformable objects in real time. This framework bridges the (self-)collision culling and detection with reduced deformable simulation using MAT. Specifically, our reduced model is constructed based on MM, which not only captures important shape deformations, but also serves as a tight volumetric envelop for CC/SCC. Our generalized coordinate compactly represents the status of each

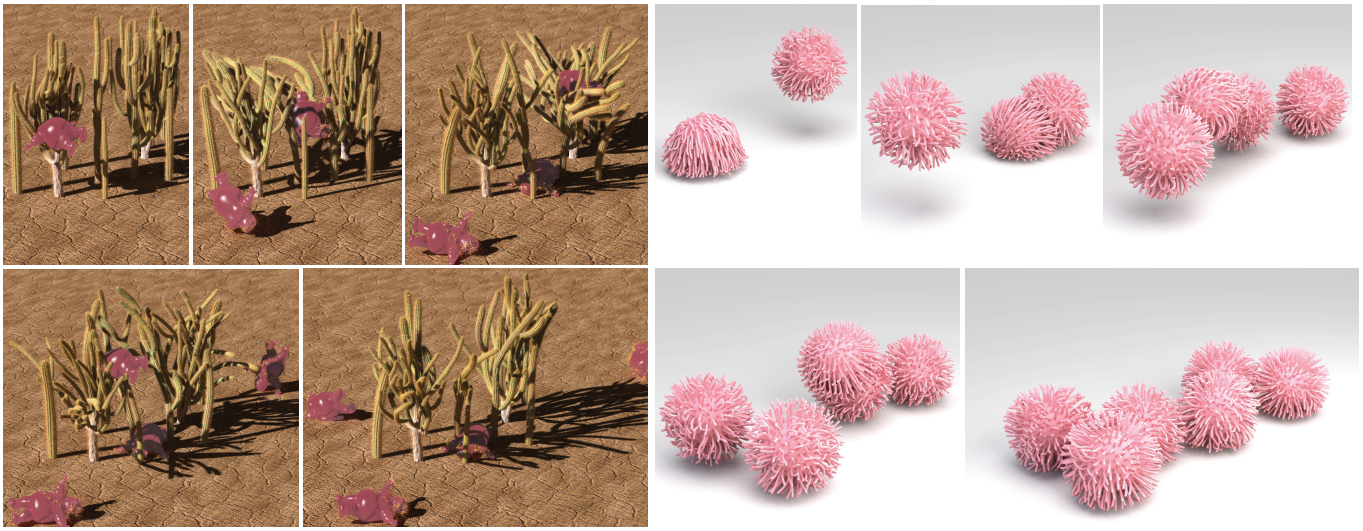

Fig. 18. **Left**: Four Staypufts fly to the cactus bush. Each Staypuft has 28 MHs, which construct a 462-dimension subspace. **Right**: Six puffer balls collide with each other. Each puffer ball embeds 1, 269 MHs, which do not only capture local deformations at its soft strings, but also help the solver quickly identify collisions and self-collisions. To ensure there are sufficient flexibilities on the body of the ball, we added extra six handles besides the MH at the center of the sphere. In this example, we also enforce the weight locality so that we can use sparse Cholesky in cuSPARSE to more efficiently solve the reduced global matrix.

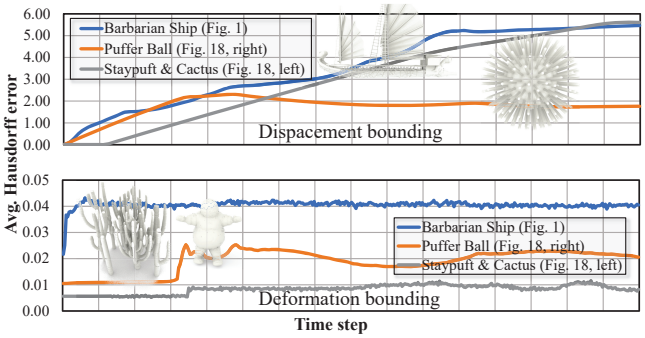

Fig. 19. Average Hausdorff errors of examples in Fig. 1 and Fig. 18. The rest-shape MAT of each model is also given. Due to the large rigid body motion in those examples, displacement bounding yields very high Hausdorff errors (e.g. over 500% of the barbarian ship in Fig. 1). The deformation bounding however produces tight enclosure of deformed model consistently.

MM, which can be directly used to update MAT for CC/SCC. We propose a subspace collision-ready matrix assembly mechanism, which keeps the reduced global matrix collision-invariant so that it can be pre-factorized. We also provide an in-depth analysis of the trade-off between local reduction and global reduction and based on which, we design a semi-reduced projective dynamics formulation. For the collision part, we propose an efficient algorithm for fast CC/SCC between MPs. If an interaction between two MPs is confirmed, we also provide a closed-form formulation to directly retrieve the location of the deepest interpenetration between MPs.

However, our system also has several limitations. First of all, the biggest advantage of this framework itself may also be its biggest weakness. Unlike [Barbič and James 2010], our system is built based on MAT, and it is less compatible with other model reduction methods. In theory, one can still use the reduced coordinate to update the configurations of the MAT, but it would be slower and less intuitive than a MAT-based model. Secondly, for the nearly spherical objects, e.g. puffer ball, we may need to add additional medial vertices manually to capture desired local deformations. Thirdly, current MAT data structure does not have any hierarchies. As a result, our method may be outperformed in collision-light animations because a simple culling at the top level of a BVH may take tens of thousands of cullings with MAT. Fortunately, we believe that this limitation can be easily fixed by mixing BVH and MAT during the CC/SCC. In the future, we plan to design multi-level MAT to address this limitation. Our subspace is constructed geometrically. Therefore, it may not be suitable for heterogeneous models. In this situation, one may consider to use different weight coefficients as in [Nesme et al. 2009] or [Faure et al. 2011] that better reflect the material variations.

REFERENCES

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. In *ACM transactions on graphics (TOG)*, Vol. 27. ACM, 165.

Baptiste Angles, Daniel Rebain, Miles Macklin, Brian Wyvill, Loic Barthe, Jp Lewis, Javier von der Pahlen, Shahram Izadi, Julien P. C. Valentin, Sofien Bouaziz, and Andrea Tagliasacchi. 2019. VIPER: Volume Invariant Position-based Elastic Rods. *PACMCGIT* 2, 2 (2019), 19:1–19:26. https://doi.org/10.1145/3340260

Andreas Antoniou and Wu-Sheng Lu. 2007. *Practical Optimization: Algorithms and Engineering Applications* (1st ed.). Springer Publishing Company, Incorporated.

Dominique Attali and Annick Montanvert. 1997. Computing and simplifying 2D and 3D continuous skeletons. *Computer vision and image understanding* 67, 3 (1997), 261–273.

David Baraff. 1994. Fast contact force computation for nonpenetrating rigid bodies. In *Computer graphics and interactive techniques*. ACM, 23–34.

Jernej Barbič and Doug James. 2007. Time-critical distributed contact for 6-dof haptic rendering of adaptively sampled reduced deformable models. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 171–180.

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

Jernej Barbič and Yili Zhao. 2011. Real-time large-deformation substructuring. In *ACM transactions on graphics (TOG)*, Vol. 30. ACM, 91.

Jernej Barbič and Doug L James. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. In *ACM Trans. Graph. (TOG)*, Vol. 24. ACM, 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), 80:1–80:13.

Robert Bridson, Ronald Fedkiw, and John Anderson. 2002. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. *ACM Trans. Graph.* 21, 3 (July 2002), 594–603.

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

Xiao-Diao Chen, Jun-Hai Yong, Guo-Qin Zheng, Jean-Claude Paul, and Jia-Guang Sun. 2006. Computing minimum distance between two implicit algebraic surfaces. *Computer-Aided Design* 38, 10 (2006), 1053–1061.

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.

Andreas Fabri and Monique Teillaud. 2011. CGAL-The Computational Geometry Algorithms Library. In *10e colloque national en calcul des structures*. 6.

Noura Faraj, Jean-Marc Thiery, and Tamy Boubekeur. 2013. Progressive medial axis filtration. In *SIGGRAPH Asia 2013 Technical Briefs*. ACM, 3.

François Faure, Benjamin Gilles, Guillaume Bousquet, and Dinesh K Pai. 2011. Sparse meshless models of complex deformable solids. In *ACM Trans. Graph. (TOG)*, Vol. 30. ACM, 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), 214:1–214:9.

Marco Fratarcangeli, Huamin Wang, and Yin Yang. 2018. Parallel iterative solvers for real-time elastic deformations. In *SIGGRAPH Asia 2018 Courses*. ACM, 14.

Ming Gao, Andre Pradhana Tampubolon, Chenfanfu Jiang, and Eftychios Sifakis. 2017. An adaptive generalized interpolation material point method for simulating elastoplastic materials. *ACM Trans. Graph. (TOG)* 36, 6 (2017), 223.

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 *Computer graphics and interactive techniques*. ACM, 171–180.

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.

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

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

Ladislav Kavan and Jiri Zara. 2005. Fast Collision Detection for Skeletally Deformable Models. *Computer Graphics Forum* 24, 3 (2005), 363–372.

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

Martin Komaritzan and Mario Botsch. 2018. Projective skinning. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 1 (2018), 12.

Shankar Krishnan, M Gopi, M Lin, Dinesh Manocha, and A Pattekar. 1998. Rapid and accurate contact determination between spline models using ShellTrees. In *Computer Graphics Forum*, Vol. 17. Wiley Online Library, 315–326.

Paul G. Kry, Doug L. James, and Dinesh K. Pai. 2002. EigenSkin: Real Time Large Deformation Character Skinning in Hardware. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '02)*. 153–159.

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.

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

Matthew Moore and Jane Wilhelms. 1988. Collision detection and response for computer animation. In *ACM Siggraph Computer Graphics*, Vol. 22. ACM, 289–298.

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

Rahul Narain, Matthew Overby, and George E. Brown. 2016. ADMM ⊇ Projective Dynamics: Fast Simulation of General Constitutive Models. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '16)*. 21–28.

Maxim Naumov. 2011. Parallel solution of sparse triangular linear systems in the preconditioned iterative methods on the GPU. *NVIDIA Corp., Westford, MA, USA, Tech. Rep. NVR-2011* 1 (2011).

Matthieu Nesme, Paul G Kry, Lenka Jeřábková, and François Faure. 2009. Preserving topology and elasticity for embedded deformable models. In *ACM Trans. Graph. (TOG)*, Vol. 28. ACM, 52.

Simon Pabst, Artur Koch, and Wolfgang Straßer. 2010. Fast and scalable cpu/gpu collision detection for rigid and deformable surfaces. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 1605–1612.

Alex Pentland and John Williams. 1989. Good vibrations: Modal dynamics for graphics and animation. In *SIGGRAPH Comput. Graph.*, Vol. 23. ACM.

Ken Shoemake. 1985. Animating rotation with quaternion curves. In *ACM SIGGRAPH computer graphics*, Vol. 19. ACM, 245–254.

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.

Svetlana Stolpner, Paul Kry, and Kaleem Siddiqi. 2012. Medial spheres for shape approximation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 6 (2012), 1234–1240.

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 (March 2016), 1278–1290.

Yun Teng, Miguel A Otaduy, and Theodore Kim. 2014. Simulating articulated subspace self-contact. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 106.

Matthias Teschner, Stefan Kimmerle, Bruno Heidelberger, Gabriel Zachmann, Laks Raghupathi, Arnulph Fuhrmann, M-P Cani, François Faure, Nadia Magnenat-Thalmann, Wolfgang Strasser, et al. 2005. Collision detection for deformable objects. In *Computer graphics forum*, Vol. 24. Wiley Online Library, 61–81.

Jean-Marc Thiery, Émilie Guy, and Tamy Boubekeur. 2013. Sphere-meshes: Shape approximation using spherical quadric error metrics. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 178.

Jean-Marc Thiery, Émilie Guy, Tamy Boubekeur, and Elmar Eisemann. 2016. Animated Mesh Approximation With Sphere-Meshes. *ACM Trans. Graph.* 35, 3, Article 30 (May 2016), 13 pages. https://doi.org/10.1145/2898350

Anastasia Tkach, Mark Pauly, and Andrea Tagliasacchi. 2016. Sphere-meshes for Real-time Hand Modeling and Tracking. *ACM Trans. Graph.* 35, 6, Article 222 (Nov. 2016), 11 pages.

Huamin Wang. 2015. A Chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Trans. Graph. (TOG)* 34, 6 (2015), 246.

Huamin Wang and Yin Yang. 2016. Descent methods for elastic body simulation on the GPU. *ACM Trans. Graph. (TOG)* 35, 6 (2016), 212.

Xinlei Wang, Min Tang, Dinesh Manocha, and Ruofeng Tong. 2018. Efficient BVH-based Collision Detection Scheme with Ordering and Restructuring. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 227–237.

Baorong Yang, Junfeng Yao, and Xiaohu Guo. 2018. DMAT: Deformable Medial Axis Transform for Animated Mesh Approximation. *Computer Graphics Forum* (2018).

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).

Yin Yang, Weiwei Xu, Xiaohu Guo, Kun Zhou, and Baining Guo. 2013. Boundary-aware multidomain subspace deformation. *IEEE transactions on visualization and computer graphics* 19, 10 (2013), 1633–1645.

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.