

Vibrational Affine Body Dynamics

HAOYANG SHI, University of Utah, USA

YUMENG HE, University of Southern California, USA

SHELDON ANDREWS, École de technologie supérieure (ÉTS), Canada

CHENFANFU JIANG, University of California, Los Angeles, USA

KUI WU, LIGHTSPEED, USA

YIN YANG, University of Utah, USA

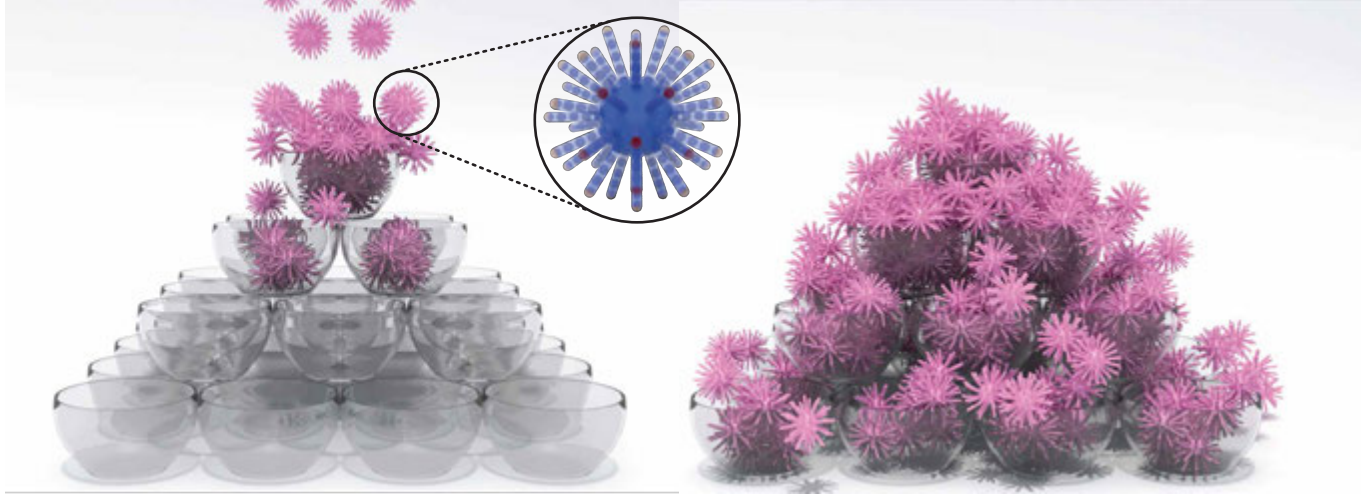


Fig. 1. Vibrational ABD. We propose vibrational affine body dynamics (VABD) to efficiently model the nonlinear dynamics of stiff but deformable objects. Vanilla RBD/ABD prescribes the body's dynamics with a low-rank displacement subspace, which fails to capture high-frequency vibrational deformation when sharp and abrupt external loads are present. VABD augments ABD with a rotation-invariant non-displacement subspace. The design of this subspace allows linear simulation of secondary vibrations, and the vibrational coordinate can be directly transformed along with the global affine motion. VABD features an interleaved medial collision mechanism, which skips broad-phase culling and significantly speeds up the collision handling. A neural medial-axis transform is proposed to establish the algorithmic linkage between the vibrational coordinate and medial DOFs. VABD also includes a novel Newton-quasi-Newton solver, aiming to track the collision Hessian efficiently while keeping the elasticity Hessian exact. The teaser figure reports a representative experiment using VABD. In this experiment, there are 356 stiff spiky balls falling into a stack of glassy bowls. The full-scale meshes consist of 5.06M tetrahedral elements and 1.7M vertices, which takes more than 100 min to simulate one time step ($h = 1/500$ s). Vibrational deformation at the spikes enables rich bouncing and collisions among the balls and with the bowls. VABD is orders of magnitude faster than stiff FEM (1,500 $\times$) or reduced FEM (46 $\times$) simulation. Each spiky ball has 12 affine modes plus 10 non-displacement vibrational modes, and it is encapsulated by 211 medial primitives, as highlighted in the figure. VABD only needs 4 s on average to simulate one step, including collision processing.

Rigid-body and affine body dynamics are default simulation modalities for rigid or near-rigid objects. Their low-rank kinematic models inherently preclude vibrational responses that stiff bodies exhibit during abrupt impacts or contacts. Simulating such behavior with full or reduced finite element method (FEM) requires high spatial resolution and small time steps, which

significantly increases computational costs of system solving and collision handling. We propose vibrational affine body dynamics (VABD), a novel simulation framework that captures these missing high-frequency vibrations while preserving excellent efficiency and robustness, even in complex scenarios. VABD introduces a carefully constructed secondary vibration subspace that augments standard affine body dynamics to encode high-frequency deformation on top of the time-varying affine frame, while remaining orthogonal to this evolving affine subspace with respect to the system matrix. This subspace design enables efficient reduced simulation with improved expressiveness. For collision handling, VABD employs medial-axis-based proxies and interleaves medial continuous and discrete collision detection to balance robustness and performance across different motion scales. We train a neural MAT (medial axis transform) mapping from reduced vibrational coordinates to medial primitives, enabling collision responses to be coupled back to the reduced system with well-defined derivatives. To efficiently solve the resulting implicit system, we introduce a Newton-quasi-Newton solver

Authors' addresses: Haoyang Shi, haoyang.shi@utah.edu, University of Utah, Salt Lake City, UT, USA; Yumeng He, heyumeng0928@gmail.com, University of Southern California, Los Angeles, CA, USA; Sheldon Andrews, sheldon.andrews@etsmtl.ca, École de technologie supérieure (ÉTS), Montréal, QC, Canada; Chenfanfu Jiang, chenfanfu.jiang@gmail.com, University of California, Los Angeles, Los Angeles, CA, USA; Kui Wu, kwu@lightspeed-studios.com, LIGHTSPEED, Los Angeles, CA, USA; Yin Yang, yangzzy@gmail.com, University of Utah, Salt Lake City, UT, USA.

Please use nonacm option or ACM Engage class to enable CC licenses. This work is licensed under a Creative Commons Attribution 4.0 International License. © 2026 Copyright held by the owner/author(s). ACM 0730-0301/2026/12-ART219 <https://doi.org/10.1145/3842507>

that exploits the fact that collision Hessians remain largely unchanged under small time steps, which effectively suppresses the overhead of repeated Hessian construction while retaining strong second-order convergence. Experiments demonstrate that VABD produces realistic vibrational responses for stiff objects undergoing intensive and extensive collisions. In terms of performance, VABD is orders of magnitude faster than FEM-based or reduced simulation approaches.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: Affine body dynamics, Neural simulation, Medial axis transform, Reduced simulation, Rigid body simulation

ACM Reference Format:

Haoyang Shi, Yumeng He, Sheldon Andrews, Chenfanfu Jiang, Kui Wu, and Yin Yang. 2026. Vibrational Affine Body Dynamics. *ACM Trans. Graph.* 45, 6, Article 219 (December 2026), 20 pages. <https://doi.org/10.1145/3842507>

1 INTRODUCTION

Rigid or near-rigid bodies constitute a core component of physically based animation and simulation. Classical rigid-body dynamics (RBD) [Baraff 1997; Bender et al. 2014] is efficient, as it evolves each object using minimal (six) degrees of freedom (DOFs). This scheme implicitly assumes *infinite stiffness* and prevents the object from storing any elastic energies. Affine body dynamics (ABD) [Lan et al. 2022] extends RBD by generalizing the model to a 12-DOF affine subspace. Added deformation DOFs allow ABD to work naturally with optimization-based collision handling methods like incremental potential contact (IPC) [Li et al. 2020]. The affine motion also enables piece-wise linear continuous collision detection (CCD). Together with IPC, ABD algorithmically guarantees that objects are free of any interpenetration.

Despite these advantages, both RBD and ABD only capture very low-frequency body dynamics. They fail to reproduce the high-frequency vibrations that emerge when stiff objects experience impulsive external loads or high-speed collisions — effects that are commonly observed for hard materials. According to Shannon’s sampling theorem [Shannon 2006], aliasing-free capture of high-frequency spatiotemporal signals requires sufficiently dense sampling in both space and time. A naïve option is to resort to high-resolution finite element methods (FEM) with large Young’s moduli to capture vibrational deformations at the cost of a (much) more expensive system solve. High-frequency vibrations also require a small time step to catch rapid strain propagation within the stiff body. Consequently, the overhead for collision detection is substantially escalated, and many efficient collision detection techniques specially tailored for rigid objects [Redon et al. 2002] are inapplicable due to the presence of vibrations.

We introduce vibrational affine body dynamics (VABD), a reduced model that captures high-frequency vibration while retaining the efficiency of affine-body simulation. VABD is built on the observation that stiff objects typically undergo large *displacement*, whereas the actual *deformation* is often of small magnitude. We design a vibration subspace that augments high-frequency deformation to the affine frame. We extract rotation-invariant components from modal poses [Pentland and Williams 1989a] to construct a different set of reduced coordinates, which allows the frame transformation to be applied directly in reduced space and makes the affine and vibrational subspaces orthogonal w.r.t. the system matrix.

Collision handling introduces an additional layer of difficulty. High-resolution geometry makes CCD expensive, and a small time step necessitates more frequent CCD invocations. To address this, we replace mesh-level primitives, namely triangles, with medial-axis primitives [Tam and Heidrich 2003] for collision queries. Medial CCD keeps the global affine deformation free of intersections. For high-frequency vibrations, we introduce medial discrete collision detection (DCD), which is significantly faster than medial CCD. During simulation, we interleave medial CCD with medial DCD so that high-frequency collisions are processed permissively, while the dominant affine motions remain penetration-free. Since collision energies are defined on the medial DOFs in medial CCD/DCD, we also need to know how the medial primitives change under vibrational deformation to compute the reduced collision forces. A missing puzzle piece is the relation between reduced DOFs and medial DOFs, for which no closed-form expression is available. To this end, we train a neural network to regress this relation in a data-driven manner, and use its derivatives to couple collision barriers back to the generalized coordinates.

On the solver side, we propose a Newton–quasi-Newton (NQN) scheme to reduce the cost of repeatedly constructing collision Hessians, exploiting the fact that collision Hessians vary gradually under small time steps. NQN solver alternates between exact Newton solves and quasi-Newton updates. Because the elasticity Hessian is inexpensive to assemble once the affine and vibrational DOFs are decoupled (due to our subspace construction modality), we design a modified L-BFGS procedure that only tracks the variation of the collision Hessian. Exact Newton solves are invoked at a lower frequency to restore full accuracy.

We evaluate the VABD framework on a broad set of experiments, and a representative one is reported in Fig. 1. Compared with RBD and ABD, VABD captures vivid high-frequency vibrations of stiff objects, especially for geometries with sharp features undergoing intensive collisions. By employing medial collision representations, VABD is substantially faster than both full FEM and reduced-order FEM, while remaining robust in collision-intensive settings.

2 RELATED WORK

In this section, we briefly review some relevant prior works. Our method resides at the intersection of rigid body simulation, deformable models, and neural network-based shape representation. Each covers a large volume of excellent contributions.

2.1 Rigid-Body Simulation

A rigid body simplifies the behavior of a highly stiff object. External stimuli such as forces or impulses propagate almost instantaneously across the object to eliminate relative deformations. This property is typically handled kinematically by directly formulating rigid body models with SE(3) coordinates. Rigid body models have been extensively studied in computer graphics since the pioneering work of [Baraff 1989, 1997]. We refer the reader to the survey by [Bender et al. 2014] for a broader overview. Since a free-floating rigid body has only six DOFs, it can be modeled very efficiently. Consequently, the primary focus of rigid body dynamics (RBD) is on

resolving collisions and contacts. To prevent object interpenetration, traditional algorithms rely on linear complementarity programming (LCP) [Baraff 1994; Stewart 2000; Trinkle et al. 2001]. These methods search for a feasible configuration within a combinatorial space, typically resolving interactions through impulses rather than forces [Baraff 1989]. However, the LCP-based contact problem is NP-hard due to the indeterminacy of which contacting nodes contribute to the collision impulse [Baraff 1991]. While velocity-level LCP approximations [Anitescu and Potra 1997; Erleben 2007; Kaufman et al. 2005] improve solvability, the resulting systems remain non-convex and difficult to solve accurately for complex scenes. Furthermore, constraint linearization in these solvers often leads to artifacts such as drifting and tunneling. While stabilization techniques [Baumgarte 1972; Cline and Pai 2003; Moreau 1988] can mitigate these issues, they may introduce new instabilities like popping. In this work, we opt for a penalty-based method [Drumwright 2007; Hasegawa et al. 2004] to handle collisions between stiff models.

2.2 Reduced Models

Deformable models concern the deformation of an object. Since the body's deformation is often much lower-rank than its actual discretizations. A common approach is to restrict the simulation to a compact subspace spanned by selected deformation modes. This reduction significantly saves computational time by reducing both the number of state variables and the size of the linear systems. Subspaces can be constructed using Linear Modal Analysis (LMA) or PCA-based methods. Barbič and James [2005]; Idelsohn and Cardona [1985] used LMA modes and their modal derivatives (MD) to build bases for constrained meshes. However, MD for free-floating bodies is potentially risky as MD bases lack *rotation invariance* and the modes may no longer cover the same deformation subspace when the object frame does not align with the frame in which the modes are computed [Benchekroun et al. 2023]. Recent work by Benchekroun et al. [2023]; Trusty et al. [2023] optimized for least-energy skinning weights rather than displacements. Each weight is driven by an affine transformation to generate displacement modes. However, their work targets secondary motions in rigged animations and does not incorporate contact. Similarly, Lan et al. [2020, 2021] used medial spheres with affine or quadratic DOFs to define reduced models. Their subspace sizes scale with the number of handles. To address rotation issues, von Tycowicz et al. [2013] added linearly transformed LMA modes to the basis. Alternatively, some methods explicitly track rotational components in reduced rotation-strain spaces [Huang et al. 2011; Li et al. 2014; Pan et al. 2015]. Beyond real-time animation [Guo et al. 2026; James and Pai 2002; Pentland and Williams 1989b], vibrational modes are widely used in sound synthesis [Langlois et al. 2014; Wang et al. 2018; Zheng and James 2011] and for modeling distant collision responses in rigid-body simulations [Coevoet et al. 2020].

2.3 Collision Culling and Collision Proxies

Collision detection is a fundamental component for modeling rigid and deformable bodies with contacts. Since exhaustive triangle-based detection is infeasible for high-resolution models, collision

culling is indispensable, typically using a bounding volume hierarchy [Meister et al. 2021]. Accelerating structures based on bounding spheres [James and Pai 2004; Zheng and James 2012] provide efficient pruning tailored for reduced simulations. Another common approach to reducing collision costs is the use of simplified geometry proxies. For example, Müller and Chentanez [2011] placed ovals as contact proxies in soft body simulations using XPBD [Macklin et al. 2016]. To reduce the total number of collision primitives, the Bullet library [Coumans 2015] generates contact proxies through convex decomposition. Similarly, nested Cages [Sacht et al. 2015] creates a coarser enveloping mesh, and Zhang et al. [2024] used a coarse mesh for contacts while ensuring it behaves in accordance with the fine mesh. However, these triangle-based methods have limitations, such as suboptimal tightness for coarse cages or requiring the fine mesh to follow specific subdivision rules. Beyond triangle meshes, sphere-Meshes [Thiery et al. 2013] use a non-manifold medial mesh to describe shapes, where vertices are spheres rather than points. On medial “cones” and “slabs” i.e., the counterparts of edges and faces in regular meshes, the distance to the central axis or plane is interpolated by the spheres, as shown in Fig. 3. Surface meshes can be converted to medial meshes via the discretized Medial Axis Transform (MAT) [Wang et al. 2024, 2022], which extracts the skeletal structure and volumetric information into sphere positions and radii. Simplifying the medial mesh after MAT is both robust and efficient [Li et al. 2016]. Even a radically simplified mesh can approximate the original shape with a low Hausdorff distance. Lan et al. [2020] were among the first to use medial representations for contact detection and handling in place of conventional triangle-based methods. This approach was later extended by Lan et al. [2021] to support CCD with penetration-free guarantees. More recently, Medial skeleton diagram [Guo et al. 2024] introduced a variant that combines the medial axis with neural-driven anisotropic distances for greater expressiveness without explicitly defining medial spheres.

2.4 Neural Shape Representations

The success of deep learning also inspires the rapid development of neural-based shape representations. While mainstream neural representations such as neural Signed Distance Fields (SDF) [Liu et al. 2022], 3D Gaussian Splatting [Kerbl et al. 2023] (3DGS), and NeRF [Mildenhall et al. 2021] are originally designed for volumetric rendering, several prior works have attempted to interface them with simulators. Most closely related to our work, Liu et al. [2024] parameterize bounding volumes using neural networks and introduce a specialized loss to avoid false negatives. Zesch et al. [2023] employ neural networks to approximate distance-based energy integrals between triangle pairs, enabling accurate and simulation-compatible collision handling. Based on 3DGS, Feng et al. [2024]; Xie et al. [2023] treated ellipsoids as particles for the MPM or SPH to bridge rendering with physics. Other approaches replace traditional mesh discretization with implicit networks. Chang et al. [2023]; Chen et al. [2023b] provided physical quantities (e.g., deformation and velocity) through neural fields, evolving reduced states by projecting sampled quantities back to a neural subspace. Neural networks have also been applied to generate reduced deformation bases. For example,

Fulton et al. [2019]; Shen et al. [2021] reduced physics-based simulation to a nonlinear neural latent space, while Modi et al. [2024] used self-supervised training to learn mesh-free skinning modes. Additionally, Chang et al. [2025] constructed neural fields representing eigen functions across a parametric shape space, and Sharp et al. [2023] learned low-order kinematic subspaces for specific meshes. However, these neural subspaces only account for the kinematics and do not directly benefit collision processing.

3 BACKGROUND

3.1 Affine Body Dynamics

We start with a brief review of ABD [He et al. 2026; Lan et al. 2022]. ABD assumes any material point i on the body undergoes a uniform linear transformation $A(t)$ and a translation $t(t)$ such that:

$$\mathbf{x}_i = A(t)\bar{\mathbf{x}}_i + \mathbf{t}(t), \quad (1)$$

where $\mathbf{x}_i$ and $\bar{\mathbf{x}}_i$ denote the current and the rest-shape position of the material point. Unlike traditional rigid body dynamics (RBD), which prescribes the rigidity kinematically i.e., $\mathbf{x}_i = \mathbf{R}(t)\bar{\mathbf{x}}_i + \mathbf{t}(t)$. ABD approximates rigid-body motion by strongly penalizing deviations from rotational transformations:

$$E_A = k_A V \|\mathbf{A}\mathbf{A}^\top - \mathbf{I}_3\|_F^2, \quad (2)$$

where $\|\cdot\|_F^2$ is the Frobenius norm of the matrix, $\mathbf{I}_3$ is a three-by-three identity matrix, k_A is the “affine stiffness” of the body, and V is the volume. The relaxation from $\mathbf{R} \in SO(3)$ to $\mathbf{A} \in \mathbb{R}^{3 \times 3}$ allows a linear subspace representation of an affine body: if a 3D model consists of N nodes, its affine deformation $\mathbf{u}_A \in \mathbb{R}^{3N}$ can be written as:

$$\mathbf{u}_A = U_A \mathbf{q}_A. \quad (3)$$

The generalized coordinate $\mathbf{q}_A = [\text{vec}^\top(\mathbf{A} - \mathbf{I}_3), \mathbf{t}^\top]^\top$ is constructed by stacking three column vectors in $\mathbf{A} - \mathbf{I}_3$ and the translation vector $\mathbf{t}$. U_A is a $3N \times 12$ matrix, and each of its three-row blocks converts the generalized coordinate $\mathbf{q}_A$ to nodal deformation:

$$\mathbf{u}_{A,i} = U_{A,i} \mathbf{q}_A = [\bar{\mathbf{x}}_i^\top \otimes \mathbf{I}_3, \mathbf{I}_3] \mathbf{q}_A, \quad (4)$$

where $\otimes$ denotes the Kronecker product. Compared with RBD, ABD features a *constant* subspace matrix U_A at the cost of an increased system size i.e., from 6 to 12.

3.2 Linear Modal Analysis

As shown in Fig. 2, after obtaining the affine basis vectors U_A , we can compute vibrational displacement modes around the rest shape i.e., $\mathbf{q}_A = \mathbf{0}_{12 \times 1}$. The vibrational modes should be kept orthogonal to U_A . This problem is formulated as finding the r lowest-energy deformation modes:

$$\Phi_V = \arg\min_{\Phi^\top M \Phi = \mathbf{I}} \text{tr}(\Phi^\top \tilde{\mathbf{K}} \Phi) \text{ s.t. } U_A^\top M \Phi = \mathbf{0}, \quad (5)$$

where $\tilde{\mathbf{K}} = H(\Psi, \mathbf{u})|_{\mathbf{u}=\mathbf{0}}$ is the rest-shape tangent stiffness matrix and M is the mass matrix. Eq. (5) can be solved as a generalized eigenvalue problem, as shown in Alg. 1.

ALGORITHM 1: Vibrational subspace construction

Input: $U_A, r, M, \tilde{\mathbf{K}}$

Output: Φ_V

```

1  $\mathbf{N} \leftarrow \text{null}(U_A^\top)$  // Compute the nullspace of  $U_A^\top$ 
2  $\tilde{\mathbf{K}} \leftarrow \mathbf{N}^\top \tilde{\mathbf{K}} \mathbf{N}$ 
3  $\tilde{\mathbf{M}} \leftarrow \mathbf{N}^\top M \mathbf{N}$ 
4  $\Phi, \Lambda \leftarrow \text{eig}(\tilde{\mathbf{K}}, \tilde{\mathbf{M}}, r)$  // Extract the  $r$  smallest eigenvalues and
   eigenvectors
5  $\Phi_V \leftarrow \mathbf{N} \Phi$ 

```

4 AUGMENT ABD WITH VIBRATION

While ABD has 6 more DOFs than RBD to express the deformation, they are responsible for uniform shape changes: three global shearing and three global scaling. Those deformations are further penalized by E_A , since ABD is designed to model (highly) stiff objects. As a result, under sharp or impulsive external forces e.g., those induced by high-speed collisions, neither RBD nor ABD can reproduce the high-frequency deformations commonly observed in the real world. This motivates us to overlay a secondary vibrational displacement field $\tilde{\mathbf{u}}_V$ on top of the affine subspace.

Fig. 2 illustrates the overview of our pipeline for augmenting ABD with vibrations. We first show that attaching a co-rotating vibrational basis to the ABD frame requires costly basis updates and introduces state-dependent derivatives (Sec. 4.1). To avoid such difficulties, we introduce a *rotation-equivariant* heterogeneous affine basis whose frame transformation can be transferred from the basis vectors to their generalized coordinates, yielding a fixed world-frame basis (Sec. 4.2). Next, we propose a simple method to convert conventional displacement modes into scalar weight modes and enforce orthogonality to the homogeneous affine basis U_A under the metric induced by the rest-state variational energy (Sec. 4.3). Finally, we leverage this orthogonality to simplify the reduced system solve and offload most of its computational cost to precomputation (Sec. 4.4).

4.1 Challenges of Frame-dependent Vibrational Basis

Overlaying a secondary vibrational displacement field $\tilde{\mathbf{u}}_V$ on top of the affine subspace, the deformed position of the material point becomes:

$$\mathbf{x}_i = A(\bar{\mathbf{x}}_i + \tilde{\mathbf{u}}_{V,i}) + \mathbf{t} = \underbrace{A\bar{\mathbf{x}}_i + \mathbf{t}}_{\text{affine displacement}} + \underbrace{A\tilde{\mathbf{u}}_{V,i}}_{\text{vibration}}. \quad (6)$$

Here, we use $[\tilde{\cdot}]$ to indicate the kinematic variable is defined in the *local affine frame*. It is converted to the world coordinate system with the current affine transformation i.e., $A\tilde{\mathbf{u}}_{V,i}$.

For a stiff object, we assume that $\tilde{\mathbf{u}}_V$ resides in a low-rank subspace of $\text{span}(\tilde{U}_V)$ such that $\tilde{\mathbf{u}}_V = \tilde{U}_V \tilde{\mathbf{q}}_V$. This leads to an alternative formulation of Eq. (6), such that

$$\mathbf{u} = U_A \mathbf{q}_A + \text{diag}_N(A) \tilde{U}_V \tilde{\mathbf{q}}_V, \quad (7)$$

where $\text{diag}_N(A) \in \mathbb{R}^{3N \times 3N}$ is a block-diagonal matrix whose three-by-three diagonal blocks are the current affine matrix:

$$A(\mathbf{q}_A) = \text{vec}^{-1}([\mathbf{I}_3, \mathbf{0}_{9 \times 3}] \mathbf{q}_A) + \mathbf{I}_3. \quad (8)$$

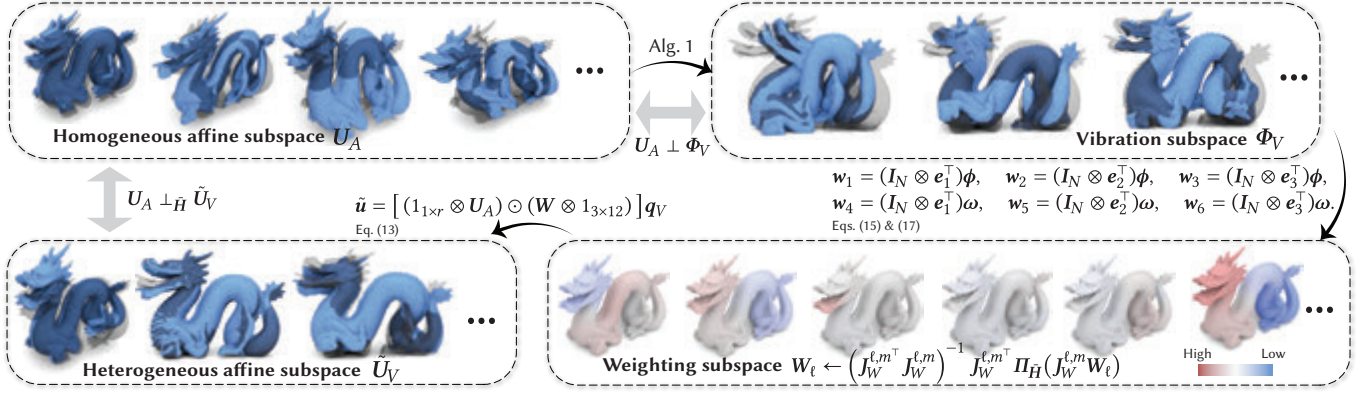


Fig. 2. Homogeneous & heterogeneous affine subspace. The vanilla ABD simulates a stiff object within the homogeneous affine subspace. On top of it, we pre-compute a group of r vibration basis vectors, extract their rotation-invariant disparities across all the nodes, and form $6r$ weighting bases of N dimensions. Those weights are used to construct a heterogeneous affine subspace, which is $\bar{H}$ -orthogonal to the homogeneous subspace.

The generalized coordinates $\tilde{q}_V$ quantify how much the body deforms along each basis direction. For general displacement modes, these coordinates are *frame-agnostic*, while the vibrational basis evolves with the affine transformation as $\tilde{U}_V = \text{diag}_N(A)\tilde{U}_V$.

Following [Bouaziz et al. 2014; Gast et al. 2015], the dynamic equilibrium of the 3D model can be formed as a variational optimization:

$$\mathbf{u}^{n+1} = \arg \min_{\mathbf{u}} E(\mathbf{u}), \quad E(\mathbf{u}) = E_I(\mathbf{u}) + \Psi(\mathbf{u}), \quad (9)$$

where Ψ is the nonlinear elasticity energy. The superscript indicates the time step index. When we use the implicit Euler time discretization, the inertia potential E_I is defined as:

$$E_I(\mathbf{u}) = \frac{1}{h^2} (\mathbf{u} - \hat{\mathbf{u}})^T \mathbf{M} (\mathbf{u} - \hat{\mathbf{u}}).$$

h is the time step size. $\mathbf{M}$ is the (lumped) mass matrix. $\hat{\mathbf{u}}$ is a known vector depending on the kinematic state of the system from the previous time step, $\hat{\mathbf{u}} = \mathbf{u}^n + h\dot{\mathbf{u}}^n + h^2\mathbf{M}^{-1}\mathbf{f}_{ext}$ with $\mathbf{f}_{ext}$ being the external force. A Newton iteration solves a $3N \times 3N$ linearized system of:

$$\mathbf{H}(E, \mathbf{u})\delta\mathbf{u} = -\mathbf{g}(E, \mathbf{u}), \quad (10)$$

where $\mathbf{H}(E, \mathbf{u}) = \frac{\partial^2 E}{\partial \mathbf{u}^2}$, and $\mathbf{g}(E, \mathbf{u}) = \frac{\partial E}{\partial \mathbf{u}}$ are the Hessian and gradient of the total variational energy E w.r.t. the unknown displacement vector $\mathbf{u}$.

Let $\tilde{\mathbf{q}} \triangleq [\mathbf{q}_A^T, \tilde{\mathbf{q}}_V^T]^T$, $\mathbf{U} \triangleq (\frac{d\mathbf{u}}{d\tilde{\mathbf{q}}})^T$. We project Eq. (10) into the subspace spanned by $\mathbf{U}$, and solve a reduced system of:

$$\mathbf{U}^T \mathbf{H} \mathbf{U} \delta\tilde{\mathbf{q}} = -\mathbf{U}^T \mathbf{g}. \quad (11)$$

While Eq. (11) is a small-size system, its assembly is costly and can be even more expensive than solving its fullspace version of Eq. (10) itself. There are three major overheads: 1) computing the full derivatives $(\frac{d\mathbf{u}}{d\tilde{\mathbf{q}}}) = [\mathbf{U}_A + \frac{\partial}{\partial \mathbf{q}_A}(\text{diag}_N(A)\tilde{\mathbf{U}}_V\tilde{\mathbf{q}}_V)^T, \text{diag}_N(A)\tilde{\mathbf{U}}_V]^T$ which couples the affine and vibrational coordinates; 2) converting the local vibrational subspace of $\mathbf{U}_V = \text{diag}_N(A)\tilde{\mathbf{U}}_V$ i.e., Eq. (7) to the world coordinate system; and 3) the subspace projection for computing the reduced Hessian $\mathbf{U}^T \mathbf{H} \mathbf{U}$ as in Eq. (11). We obviate these difficulties by using a different set of *frame-invariant* basis,

which allows the transformation $\text{diag}_N(A)$ to be directly applied at $\mathbf{q}_V$ rather than basis vectors $\tilde{\mathbf{U}}_V$.

4.2 Homogeneous & Heterogeneous Affine Subspace

The vanilla ABD imposes a uniform affine field over the model i.e., all the nodes share the same $\mathbf{q}_A$ per Eq. (3). We therefore refer to $\text{span}(\mathbf{U}_A)$ as a homogeneous affine subspace. Under this intuition, secondary vibrations emerge as a natural consequence of affine fields acting *non-uniformly*. This is analogous to a skinning decomposition, where the affine fields correspond to the bone transformations that control the local deformation using non-uniform skinning weights.

For an affine transformation A_j , we model its non-uniformity at the i -th node by scaling A_j with a varying weight coefficient w_{ij} so that the local nodal vibration $\tilde{\mathbf{u}}_i$ is the superposition of r non-uniform affine fields:

$$\tilde{\mathbf{u}}_i = \sum_{j=1}^r w_{ij} ((A_j - I_3)\tilde{\mathbf{x}}_i + \mathbf{t}_j). \quad (12)$$

We form a weighting matrix $\mathbf{W} \in \mathbb{R}^{N \times r}$ collecting w_{ij} at all the N nodes and r affine fields, and a vibrational deformation over the entire object can be re-written as:

$$\tilde{\mathbf{u}} = \underbrace{[(1_{1 \times r} \otimes \mathbf{U}_A) \odot (\mathbf{W} \otimes 1_{3 \times 12})]}_{\tilde{\mathbf{U}}_V} \tilde{\mathbf{q}}_V, \quad (13)$$

where $\odot$ denotes the Hadamard product. It is worth noting that the generalized coordinate $\tilde{\mathbf{q}}_V \in \mathbb{R}^{12r}$ in Eq. (13) has clear geometric meaning: it stores the coefficients of all affine fields $[\text{vec}^T(A_j - I_3), \mathbf{t}_j^T]^T$. Using Eqs. (12) and (13), we can verify that the scalar weight field $\mathbf{W}$ enables the desired property of rotational equivariance:

$$\mathbf{U}_V(A)\tilde{\mathbf{q}}_V = \text{diag}_N(A)\tilde{\mathbf{U}}_V\tilde{\mathbf{q}}_V = \tilde{\mathbf{U}}_V\text{diag}_{4r}(A)\tilde{\mathbf{q}}_V. \quad (14)$$

If we define the world-frame vibrational coordinates as

$$\mathbf{q}_V \triangleq \text{diag}_{4r}(A)\tilde{\mathbf{q}}_V, \quad \mathbf{q} \triangleq [\mathbf{q}_A^T, \mathbf{q}_V^T]^T, \quad (15)$$

Using these coordinates, Eqs. (1) and (11) in the object frame are substituted by:

$$\mathbf{u} = \mathbf{U}_A \mathbf{q}_A + \tilde{\mathbf{U}}_V \mathbf{q}_V, \quad (16)$$

$$\tilde{\mathbf{U}}^\top \mathbf{H} \tilde{\mathbf{U}} \delta \mathbf{q} = -\tilde{\mathbf{U}}^\top \mathbf{g}. \quad (17)$$

In this form, the affine transform is absorbed into the generalized coordinates, and the basis is independent of the affine frame.

4.3 Converting Displacement Modes to Weight Modes

Our next task is to reshape the previously obtained vibrational displacements via Alg. 1 into the form of Eq. (13). In particular, we seek to compute $\mathbf{W}$ such that the resulting $\tilde{\mathbf{U}}_V$ spans a subspace that closely matches the one produced by Alg. 1, namely $\text{span}(\Phi_V)$.

For a given displacement mode $\phi \in \mathbb{R}^{3N}$ i.e., a column vector in Φ_V , we measure the node-wise non-uniformity to create six N -dimensional weighting vectors. The first three bases describe the *translational variations* in x , y , and z directions. Therefore, they can be conveniently built as:

$$\mathbf{w}_1 = (\mathbf{I}_N \otimes \mathbf{e}_1^\top) \phi, \quad \mathbf{w}_2 = (\mathbf{I}_N \otimes \mathbf{e}_2^\top) \phi, \quad \mathbf{w}_3 = (\mathbf{I}_N \otimes \mathbf{e}_3^\top) \phi, \quad (18)$$

where we have $\mathbf{e}_1 = [1, 0, 0]^\top$, $\mathbf{e}_2 = [0, 1, 0]^\top$, and $\mathbf{e}_3 = [0, 0, 1]^\top$ respectively.

The other three bases focus on the *rotational variations* in ϕ . To this end, we compute the *curl* of the displacement field as:

$$\omega = \nabla \times \phi, \quad (19)$$

and compute $\mathbf{w}_4$, $\mathbf{w}_5$, and $\mathbf{w}_6$ in a similar way:

$$\mathbf{w}_4 = (\mathbf{I}_N \otimes \mathbf{e}_1^\top) \omega, \quad \mathbf{w}_5 = (\mathbf{I}_N \otimes \mathbf{e}_2^\top) \omega, \quad \mathbf{w}_6 = (\mathbf{I}_N \otimes \mathbf{e}_3^\top) \omega. \quad (20)$$

Intuitively, $\nabla \times$ extracts the instantaneous rotation at each node induced by ϕ . This local rotation can be approximated as $\mathbf{I}_3 + [\omega_i]_\times$, where $[\omega_i]_\times \in \text{so}(3)$ is the skew symmetric matrix of ω_i at the i -th node. It is computed as the volume-weighted average of the curls at its adjacent elements:

$$\omega_i = \frac{\sum_k V_k \omega_k}{\sum_k V_k}, \quad [\omega_k]_\times = \frac{1}{2}(\mathbf{F}_k - \mathbf{F}_k^\top), \quad (21)$$

where V_k is the rest-shape volume of an incident element k to the node whose deformation gradient is $\mathbf{F}_k$. That said, we explicitly ignore three shearing and three stretching deformations when building $\mathbf{W}$. This is because these deformation modes are strongly discouraged by E_A in Eq. (2). Eqs. (18) and (20) together deliver 6r weighting bases, and each basis is of N dimension. We apply Principal Component Analysis (PCA) to extract the first r most important weights to assemble $\mathbf{W}$.

With some manipulation of Eq. (13), it is easy to verify that a single weighting basis $\mathbf{W}_\ell \in \mathbb{R}^N$ ($\ell = 1, \dots, r$) expands twelve $3N$ -dimension displacement modes such that:

$$\text{vec}(\tilde{\mathbf{U}}_V^\ell) = \mathbf{J}_W^\ell \mathbf{W}_\ell, \quad (22)$$

where $\tilde{\mathbf{U}}_V^\ell \in \mathbb{R}^{3N \times 12}$ consists of the resulting displacement modes. $\mathbf{J}_W^\ell \in \mathbb{R}^{36N \times N}$ is a constant Jacobi matrix since $\tilde{\mathbf{U}}_V$ depends linearly on $\mathbf{W}$, and its closed form can be derived as:

$$\mathbf{J}_W^\ell = \mathcal{D}(\text{vec}(\mathbf{U}_A))(\mathbf{I}_N \otimes \mathbf{1}_{36 \times 1}), \quad (23)$$

where $\mathcal{D}(\cdot)$ stands for the diagonal matrix constructed from its vector argument. Note that $\mathcal{D}(\cdot)$ should not be confused with $\text{diag}_k(\cdot)$,

which we use for block-diagonal matrices. We further extract each of the twelve $3N$ -row in $\mathbf{J}_W^\ell$ and denote them as $\mathbf{J}_W^{\ell, m}$ for $m = 1, \dots, 12$, so that $\mathbf{J}_W^{\ell, m} \mathbf{W}_\ell$ maps $\mathbf{W}_\ell$ to the m -th column in the displacement modes $\tilde{\mathbf{U}}_V^\ell$. This allows us to perform the generalized Gram-Schmidt process so that $\mathbf{W}$ leads to displacement modes that are $\tilde{\mathbf{H}}$ -orthogonal to $\mathbf{U}_A$ i.e., $\mathbf{U}_A^\top \tilde{\mathbf{H}} \mathbf{J}_W^{\ell, m} \mathbf{W}_\ell = \mathbf{0}_{12 \times 1}$, where $\tilde{\mathbf{H}}$ is the rest-shape Hessian of the total variational energy. Mathematically, it is achieved by:

$$\mathbf{W}_\ell \leftarrow \left(\mathbf{J}_W^{\ell, m^\top} \mathbf{J}_W^{\ell, m} \right)^{-1} \mathbf{J}_W^{\ell, m^\top} \Pi_{\tilde{\mathbf{H}}} \left(\mathbf{J}_W^{\ell, m} \mathbf{W}_\ell \right), \quad (24)$$

where the operator $\Pi_{\tilde{\mathbf{H}}}$ is defined as:

$$\Pi_{\tilde{\mathbf{H}}}(\mathbf{a}) = \left(\mathbf{I}_{3N} - \mathbf{U}_A (\mathbf{U}_A^\top \tilde{\mathbf{H}} \mathbf{U}_A)^{-1} \mathbf{U}_A^\top \tilde{\mathbf{H}} \right) \mathbf{a}. \quad (25)$$

As we will see later, $\tilde{\mathbf{H}}$ -orthogonality between $\mathbf{U}_A$ and $\tilde{\mathbf{U}}_V$ decouples the affine deformation and secondary vibrations during the simulation.

Discussion. The proposed heterogeneous affine subspace closely resembles Fast Complementary Dynamics (FCD) [Benchekroun et al. 2023]. FCD was originally developed for skinning-based deformations driven by rig animations. Interestingly, ABD can be interpreted as a special case of single-joint rigging with a uniform skinning weight of $\mathbf{W}_A = \mathbf{1}$. Therefore, it is not a coincidence that Eq. (13) has the same form as FCD modal bases. Under this perspective, one could analogously construct a weighting subspace, similar to FCD, to capture local vibrational behavior. However, FCD requires specifying a probability distribution over affine coordinates and solving an N -dimensional optimization problem for each column of $\mathbf{W}$. In contrast, our formulation is more flexible: we provide a closed-form solution for computing $\mathbf{W}$ for any given displacement modes. As a result, our approach is compatible with most existing subspace construction pipelines.

4.4 Reduced Simulation

With homogeneous and heterogeneous affine subspaces, our reduced Hessian i.e., see Eq. (17) can be arranged block-wisely as:

$$\tilde{\mathbf{U}}^\top \mathbf{H} \tilde{\mathbf{U}} = \begin{bmatrix} \mathbf{U}_A^\top \\ \tilde{\mathbf{U}}_V^\top \end{bmatrix} \mathbf{H} \begin{bmatrix} \mathbf{U}_A & \tilde{\mathbf{U}}_V \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{AA} & \mathbf{H}_{AV} \\ \mathbf{H}_{AV}^\top & \mathbf{H}_{VV} \end{bmatrix}. \quad (26)$$

Its two diagonal blocks are:

$$\mathbf{H}_{AA} = \mathbf{U}_A^\top \mathbf{H} \mathbf{U}_A, \quad \mathbf{H}_{VV} = \tilde{\mathbf{U}}_V^\top \mathbf{H} \tilde{\mathbf{U}}_V. \quad (27)$$

From the equivariance property Eq. (14), for any non-singular linear transformation $\mathbf{A}$, the relation $\tilde{\mathbf{U}}_V \tilde{\mathbf{q}}_V = \text{diag}_N(\mathbf{A}^{-1}) \tilde{\mathbf{U}}_V \text{diag}_{4r}(\mathbf{A}) \tilde{\mathbf{q}}_V$ holds for any $\tilde{\mathbf{q}}_V$. This means:

$$\tilde{\mathbf{U}}_V = \text{diag}_N(\mathbf{A}^{-1}) \tilde{\mathbf{U}}_V \text{diag}_{4r}(\mathbf{A}). \quad (28)$$

The same property holds for $\mathbf{U}_A$:

$$\mathbf{U}_A = \text{diag}_N(\mathbf{A}^{-1}) \mathbf{U}_A \text{diag}_4(\mathbf{A}). \quad (29)$$

Meanwhile, by neglecting the influence of vibrational deformation on the Hessian, the Hessian can be approximated from the warped rest-shape Hessian $\tilde{\mathbf{H}}$ [Müller et al. 2002]:

$$\mathbf{H}(\mathbf{A}) \approx \text{diag}_N(\mathbf{A}) \tilde{\mathbf{H}} \text{diag}_N(\mathbf{A}^{-1}) \approx \text{diag}_N(\mathbf{A}) \tilde{\mathbf{H}} \text{diag}_N(\mathbf{A}^\top). \quad (30)$$

The approximation error in Eq. (30) vanishes as $E_A \rightarrow 0$ i.e., when $\mathbf{A}$ approaches a rotation matrix. The error decays proportionally to $\frac{1}{k_A}$, which is negligible in practice since k_A is typically very large for stiff materials. Eqs. (28) and (30) allows us to simplify H_{VV} based on the rest-shape Hessian $\tilde{H}$:

$$\begin{aligned} H_{VV} &= \tilde{U}_V^\top \tilde{H} \tilde{U}_V \\ &\approx \tilde{U}_V^\top \text{diag}_N(\mathbf{A}) \tilde{H} \text{diag}_N(\mathbf{A}^\top) \tilde{U}_V \\ &= \text{diag}_{4r}(\mathbf{A}) \tilde{U}_V^\top \tilde{H} \tilde{U}_V \text{diag}_{4r}(\mathbf{A}^\top). \end{aligned} \quad (31)$$

Bearing the orthogonality between $\mathbf{U}_A$ and $\tilde{U}_V$ and Eqs. (29) and (28) in mind, the off-diagonal block H_{AV} can also be simplified to:

$$\begin{aligned} H_{AV} &= \mathbf{U}_A^\top \tilde{H} \tilde{U}_V \\ &= \text{diag}_4(\mathbf{A}) \mathbf{U}_A^\top \text{diag}_N(\mathbf{A}^{-1}) \tilde{H} \text{diag}_N(\mathbf{A}) \tilde{U}_V \text{diag}_{4r}(\mathbf{A}^{-1}) \\ &\approx \text{diag}_4(\mathbf{A}) \mathbf{U}_A^\top \tilde{H} \tilde{U}_V \text{diag}_{4r}(\mathbf{A}^{-1}) = \mathbf{0}_{12 \times 12r}. \end{aligned} \quad (32)$$

The top-left Hessian block H_{AA} describes homogeneous affine dynamics. For this sub-system, we replace Ψ with E_A to lower the computation cost for the reduced Hessian:

$$H_{AA} = \mathbf{U}_A^\top \mathbf{H}(E_I + \Psi, \mathbf{u}) \mathbf{U}_A \approx \mathbf{U}_A^\top \mathbf{H}(E_I + E_A, \mathbf{u}) \mathbf{U}_A. \quad (33)$$

From the performance perspective, the middle term in Eq. (31) $\tilde{H}_{VV} \triangleq \tilde{U}_V^\top \tilde{H} \tilde{U}_V \in \mathbb{R}^{4r \times 4r}$ can be precomputed, and the rest of the matrix products can be computed at $O(r^2)$ cost. Importantly, the full Hessian of $\mathbf{H}$ of the material model Ψ is only used for generating $\tilde{H}_{VV}$, and it is never explicitly assembled during the simulation.

5 ASYNCHRONOUS COLLISION DETECTION USING NEURAL MAT

VABD augments ABD with a heterogeneous affine subspace to accommodate high-frequency vibrations. Solving these vibrations with numerical methods requires sufficiently fine spatial and temporal discretizations, per the Nyquist-Shannon sampling theorem. Specifically, the stiff object must have a mesh resolution that is capable of describing the localized deformation without aliasing, and the time step must also be small enough to match fast-evolving shape changes. These requirements impose significant challenges for simulation performance. A high mesh resolution increases the cost of collision detection and resolution, and this cost is further up-scaled when a smaller time step is used. This largely stultifies our earlier efforts at using reduced simulation for VABD. Hence, efficiently processing collisions and collision response becomes the key issue to resolve.

Our strategy is to employ different collision detection/resolution schemes for deformations at different levels of detail. We ensure that the homogeneous ABD remains free of interpenetration, so that users do not observe catastrophic tunneling or overlap artifacts. Meanwhile, vibration-induced high-frequency collisions are processed permissively without strict guarantees, in exchange for improved performance. Specifically, we use CCD for homogeneous affine deformations and DCD for high-frequency heterogeneous affine deformations. The prior knowledge of small deformation allows us to skip triangle-level collision detection with MAT-based shape proximity for collision processing.

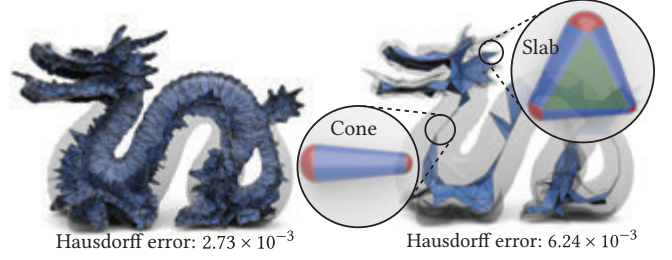


Fig. 3. Medial axis transformation. By exploiting the small-deformation prior, we use MAT as the reduced representation for collision processing, which well encloses the rest-shape model with a handful of MPs, namely cones and slabs. The distance between MPs is a high-order polynomial of centers/radii of medial spheres.

5.1 Medial Axis Transform

The medial axis (MA) of a 3D model is a set of centers of maximally-inscribed spheres having at least two closest points on its boundary [Choi et al. 1997]. By incorporating the radius information along the MA, there exists a *unique* medial axis transform (MAT) representing the original 3D model losslessly. As the MAT is a volumetric descriptor, one can easily perform the in/out test by comparing the distance to the nearest medial point and the corresponding radius. An MA can be further simplified to a 2D non-manifold triangle mesh or medial mesh e.g., see [Sun et al. 2015]. The radius information is interpolated among two (across an edge) or three (across a triangle) medial vertices on the medial mesh, creating the so-called medial cones and medial slabs. Those medial primitives (MPs) become the new surrogates of the shape discretization.

We use Q-MAT [Li et al. 2015] for medial mesh simplification. Initially, the edges on the medial mesh are organized in a priority queue according to a metric measuring the approximation error if the corresponding edge were merged into a single vertex. We iteratively collapse the edge at the top of the priority queue and update the error metric. Afterwards, we verify that all the surface nodes of the model are within the MAT volume by increasing the medial radii correspondingly. Even under aggressive simplification, the resulting MAT volume still captures the 3D shape with small Hausdorff errors, as shown in Fig. 3.

5.2 Medial CCD & Medial DCD

Any collision detection algorithm is built upon a distance function that evaluates the closest distance between surface primitives. For triangle-level collision detection, this boils down to point-triangle and edge-edge distances. Given a pair of MPs e.g., two cones C_1 and C_2 , the closest Euclidean distance between them is:

$$\min_{\alpha, \beta} \{ \|c_1(\alpha) - c_2(\beta)\| - (R_1(\alpha) + R_2(\beta)) \},$$

where $\alpha, \beta \in [0, 1]$ are two interpolation parameters. Let $c_{1,1}$ and $c_{1,2}$ be the centers of two end spheres on C_1 whose radii are $R_{1,1}$ and $R_{1,2}$. Similarly, $c_{2,1}$ and $c_{2,2}$ are the centers of two end spheres on C_2 with radii being $R_{2,1}$ and $R_{2,2}$. Both sphere center and radius are interpolated linearly along C_1 and C_2 such that $c_1(\alpha) = \alpha c_{1,1} + (1 - \alpha)c_{1,2}$, $R_1(\alpha) = \alpha R_{1,1} + (1 - \alpha)R_{1,2}$, and $c_2(\beta) = \beta c_{2,1} + (1 - \beta)c_{2,2}$,

$R_2(\beta) = \beta R_{2,1} + (1 - \beta)R_{2,2}$. Following the strategy in [Lan et al. 2021], we adopt the modified *medial distance* metric:

$$d(C_1, C_2) = \min_{\alpha, \beta} \|c_1(\alpha) - c_2(\beta)\|^2 - (R_1(\alpha) + R_2(\beta))^2, \quad (34)$$

We refer the reader to [Lan et al. 2021] for the detailed derivation of the slab–cone and slab–slab distance gradient and hessian formulas, as well as special safeguards for several corner cases.

5.2.1 Medial CCD. Just as in triangle-level CCD, medial CCD aims to find the time of impact (TOI) between a pair of MPs, namely cone–cone, cone–slab, or slab–slab based on the distance metric defined in Eq. (34). This leads to a root-finding problem involving high-order polynomials. For example, slab–slab CCD yields a sextic polynomial in t , which we solve numerically using Newton’s method. The cost of finding the t_{TOI} for a pair of MPs is certainly higher than that of point–triangle or edge–edge CCD. Nevertheless, medial CCD remains much more efficient than its triangle-level counterpart because MAT-based shape proxies generate far fewer primitive pairs for fine models (which are needed for capturing high-frequency vibrations). As a result, a broad-phase collision culling stage is no longer necessary. Based on the output of medial CCD, a logarithmic IPC barrier energy [Li et al. 2020] is inserted into the global variational optimization (Eq. (9)) if the medial distance between two MPs is smaller than a user-specific threshold $\hat{d}$:

$$B_{ccd}(d, \hat{d}) = \begin{cases} -\kappa_{ccd}(d - \hat{d})^2 \ln\left(\frac{d}{\hat{d}}\right), & 0 < d \leq \hat{d}, \\ 0, & d > \hat{d}. \end{cases} \quad (35)$$

5.2.2 Medial DCD. Plausible simulation of high-frequency vibrations also requires sufficient temporal resolution i.e., a small time step size. However, invoking CCD, even medial CCD at a high cadence still brings substantial overheads and is often unnecessary for many practical applications. For a better performance–accuracy trade-off, we interleave medial CCD with medial DCD, a more efficient collision processing modality to complement medial CCD.

Medial DCD checks p^* for all pairs of MPs (excluding adjacent primitives on the MA) at the end of a time step. We activate an *exponential barrier* whenever DCD returns a negative or close-to-zero distance measure:

$$B_{dcd}(d, \hat{d}) = \begin{cases} \kappa_{dcd}(d - \hat{d})^2, & 0 \leq d < \hat{d}, \\ \kappa_{dcd} \left(e^{-d} - 1 + d + \frac{1}{2}d^2 - 2\hat{d}d + \hat{d}^2 \right), & d < 0. \end{cases} \quad (36)$$

Intuitively, Eq. (36) serves as a regular quadratic repulsion when two MPs are close to each other ($d < \hat{d}$), and it strongly penalizes overlapping MPs when $d < 0$. Both Eqs. (35) and (36) are C^2 -continuous. The difference lies in the domains of those two barrier functions. IPC barrier is undefined when $d < 0$. Therefore, it must be paired with CCD-based line search filtering to keep all MP pairs separate. When medial DCD is used, B_{dcd} remains well-defined with a negative distance, and $B_{dcd} \rightarrow \infty$ as the penetration depth increases. Appendix A shows the full derivation of gradient and Hessian of the collision barrier functions B_{ccd} and B_{dcd} w.r.t. generalized coordinates for implicit time integration.

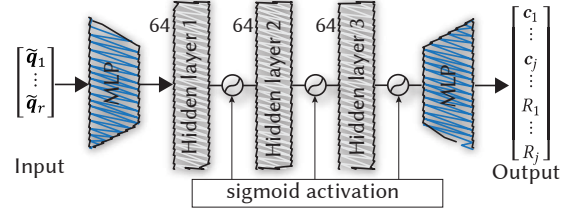


Fig. 4. Neural MAT. We train a compact neural network to implicitly regress the updated configuration of MAT (p) given a heterogeneous affine deformation $\tilde{q}_V$.

5.3 Neural MAT

Building a high-quality MAT of a given 3D model is expensive, and the state of an MAT varies under different deformation poses. It is clearly infeasible to generate an MAT from scratch at each time step. Our goal is to obtain the updated MAT configuration directly through the reduced coordinate q .

Let us use $p = [c_1^T, \dots, c_j^T, \dots, R_1, \dots, R_j, \dots]^T \in \mathbb{R}^{4m}$ to denote the state vector of the MAT with m medial spheres. The center and radius of the j -th medial sphere are c_j and R_j , and its rest-shape center and radius are $\bar{c}_j$ and $\bar{R}_j$, respectively. If the object only exhibits a homogeneous affine deformation i.e., as defined in Eq. (1), the updated state of the medial sphere can be obtained as:

$$c_j(q_A) = A\bar{c}_j + t, \quad R_j(q_A) \approx \|A\|_2 \bar{R}_j \approx \bar{R}_j. \quad (37)$$

Here, $\|A\|_2$ is the matrix spectral norm, and it nearly equals one under the stiff object assumption.

The presence of heterogeneous affine deformation makes updating the MAT non-trivial as p nonlinearly depends on the object-frame vibrational coordinates $\tilde{q}_V$, and a closed-form formulation of $p(\tilde{q}_V)$ is not available. Our answer to this difficulty is to train a light-weight neural network to learn the mapping between $\tilde{q}_V$ and p to facilitate the collision process.

Fig. 4 visualizes the architecture of the network we used. It takes $\tilde{q}_V$ as the input, which concatenates $\tilde{q}_{Vj}$ for all r heterogeneous affine fields. An MLP, i.e., an encoder module, transfers the input to three internal hidden layers. Each hidden layer contains 64 neurons, and they are nonlinearly activated with *sigmoid*. Finally, it is decoded with another MLP to the corresponding medial state vector $p(\tilde{q}_V)$. We note that the activation function is chosen over other popular options e.g., *ReLU* to ensure the differentiability of the network so that $\frac{\partial p}{\partial \tilde{q}_V}$ are well defined. We derive those network derivatives in Appendix B.

As shown in Sec. 7, neural MAT predicts a near-optimal configuration of MAT under reduced deformations. It substantially improves the quality of the deformed MAT compared with the existing method [Lan et al. 2020], which updates medial spheres via embedded barycentric interpolation on the tetrahedral mesh. This higher-fidelity MAT leads to more reliable medial DCD and, consequently, better convergence behavior in collision-intensive scenarios.

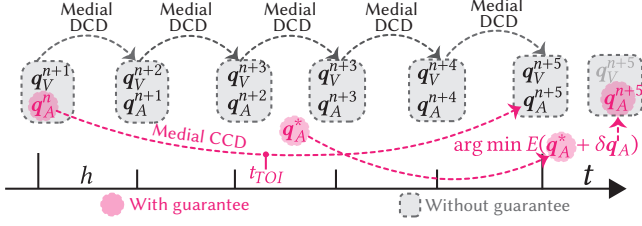


Fig. 5. Interleaved collision processing. To balance speed and accuracy, we employ a hybrid collision-processing scheme: medial DCD handles most time steps and provides robust repulsion, while medial CCD is triggered at a coarser interval to keep the homogeneous affine deformation free of interpenetration.

5.4 Interleaved Collision Pipeline

Our collision processing scheme alternates between medial CCD and medial DCD. Specifically, medial CCD handles the deformation induced by $\mathbf{q}_A$, while medial DCD is in charge of the secondary deformation encoded in $\mathbf{q}_V$. Because the time step h is small, we periodically launch medial CCD to ensure that the underlying homogeneous affine deformation remains free of interpenetration over a wider temporal window. In our implementation, medial CCD is often executed every 5 time steps with a default time step size of $h = 1/500$ sec, which corresponds to a CCD invocation frequency of 100 Hz.

Medial CCD ensures $\mathbf{q}_A$ is penetration-free. To this end, we update the rest-shape MAT according to the homogeneous affine deformation using Eq. (37) while ignoring $\mathbf{q}_V$. In other words, neural MAT is not used here. As shown in Fig. 5, a medial CCD routine calculates t_{TOI} based on the most recent penetration-free $\mathbf{q}_A^n$ and the output from the last DCD step $\mathbf{q}_A^{n+5}$. If the resulting TOI is valid and between $(0, 1]$, we obtain an updated safe affine displacement $\hat{\mathbf{q}}_A \leftarrow \mathbf{q}_A^n + 0.9 t_{TOI}(\mathbf{q}_A^{n+5} - \mathbf{q}_A^n)$. Re-integrating the current step using $\hat{\mathbf{q}}_A$ as the initial guess within $\text{span}(U_A)$ resolves model intersections in $\mathbf{q}_A^{n+5}$. This strategy seamlessly works with decoupled homogeneous and heterogeneous affine dynamics — $\mathbf{q}_A$ does not influence the value of $\mathbf{q}_V$ as the reduced Hessian is block-diagonal.

Medial DCD is applied at most time steps. It evaluates MP–MP distances under the heterogeneous affine deformation $\mathbf{p}(\mathbf{q}_V)$ predicted by the neural MAT (Fig. 4) and contributes the corresponding B_{dcd} to Eq. (9). Medial DCD runs efficiently on the GPU i.e., it is about $20\times$ faster than medial CCD and $200\text{--}500\times$ faster than triangle-level CCD. Although medial DCD does not provide a strict penetration-free guarantee, in practice it generates repulsion strong enough to keep objects separated with a negligible cost. The low-frequency motion governed by $\mathbf{q}_A$ is handled using medial CCD at wider intervals, which is adequate because such coarse-scale deformation does not require high-frequency collision enforcement and remains visually pleasant even when CCD is invoked less often.

Discussion. We emphasize that this mixed CCD–DCD scheme is *not* the only collision handling strategy for VABD. Instead, it is a deliberate compromise among simulation performance, visual plausibility, and algorithmic exactness, and it is designed to apply collision methods that best suit dynamics at different frequency

scales. If desired, one may stick with triangle-level CCD at every time step for a strictly penetration-free result, as in [Ferguson et al. 2021; Lan et al. 2022], or revert entirely to a DCD-based approach such as [Chen et al. 2023a].

6 NQN SOLVER

As discussed in Sec. 4.4, the Hessian of a collision-free VABD system is block-diagonal: $\mathbf{H}_{VV}$ is computed via Eq. (31) with $O(r^2)$ time, and the assembly cost of $\mathbf{H}_{AA}$ only scales linearly with the number of bodies. Since r is small (typically $r = 10$), factorizing the VABD system matrix is also efficient. In contrast, constructing the barrier Hessian $\mathbf{H}(B, \mathbf{q})$ is the dominant cost in the simulation. This step requires integrating $\frac{\partial d}{\partial \mathbf{p}}$ and $\frac{\partial^2 d}{\partial \mathbf{p}^2}$ for all MP pairs and projecting them to the tangent space defined by $\frac{\partial \mathbf{p}}{\partial \mathbf{q}}$.

We follow the same idea as our interleaved collision-processing scheme to lower the cost of constructing the barrier Hessian by using an NQN solver that alternates Newton and quasi-Newton steps. When medial CCD is executed, we solve the system with the exact Hessian of $\mathbf{H}(E + B, \mathbf{q})$ using Newton’s method. At the remaining time steps, we apply the L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) [Nocedal and Wright 2006] for medial DCD.

L-BFGS keeps a brief history of $\mathbf{q}_{(k)}$ and $\mathbf{g}_{(k)}$ from previous iterations to infer the curvature information through the secant condition:

$$\mathbf{H}_{(k+1)} (\mathbf{q}_{(k+1)} - \mathbf{q}_{(k)}) = \mathbf{g}_{(k+1)} - \mathbf{g}_{(k)}. \quad (38)$$

Here, we use the parenthesized subscript (k) to denote the iteration index. $\mathbf{g}_{(k+1)} = \mathbf{g}_{(k+1)}(E + B, \mathbf{q})$ and $\mathbf{H}_{(k+1)} = \mathbf{H}_{(k+1)}(E + B, \mathbf{q})$ are the gradient and Hessian of the elasticity and collision barrier. The new Hessian inverse is approximated via:

$$\mathbf{H}_{(k+1)}^{-1} = \left(\mathbf{I} - \frac{\mathbf{s}_{(k)} \mathbf{y}_{(k)}^\top}{\mathbf{y}_{(k)}^\top \mathbf{s}_{(k)}} \right) \mathbf{H}_{(k)}^{-1} \left(\mathbf{I} - \frac{\mathbf{s}_{(k)} \mathbf{y}_{(k)}^\top}{\mathbf{y}_{(k)}^\top \mathbf{s}_{(k)}} \right) + \frac{\mathbf{s}_{(k)} \mathbf{s}_{(k)}^\top}{\mathbf{y}_{(k)}^\top \mathbf{s}_{(k)}}, \quad (39)$$

where we have $\mathbf{s}_{(k)} = \mathbf{q}_{(k+1)} - \mathbf{q}_{(k)}$, and $\mathbf{y}_{(k)} = \mathbf{g}_{(k+1)} - \mathbf{g}_{(k)}$. As mentioned before, neither matrix factorization nor the assembly of elasticity Hessian is our system bottleneck. Therefore, We use the L-BFGS procedure for tracking the variation of the barrier Hessian.

To this end, we collect the history of barrier gradient $\mathbf{g}_{(k)}(B, \mathbf{q})$ and denote $\mathbf{y}_{(k)}(B) = \mathbf{g}_{(k+1)}(B, \mathbf{q}) - \mathbf{g}_{(k)}(B, \mathbf{q})$. The secant condition of the barrier Hessian is:

$$\mathbf{H}_{(k+1)}(B) \mathbf{s}_{(k)} = \mathbf{y}_{(k)}(B). \quad (40)$$

At the $(k+1)$ -th iteration, we build the elasticity Hessian $\mathbf{H}_{(k+1)}(E)$ exactly so that Eq. (38) can be re-written as:

$$\mathbf{H}_{(k+1)} \mathbf{s}_{(k)} = \mathbf{y}_{(k)}(B) + \mathbf{H}_{(k+1)}(E) \mathbf{s}_{(k)}. \quad (41)$$

The proposed NQN algorithm stores the history of $\tilde{\mathbf{y}}_{(k)}$, which is the r.h.s. of Eq. (41) such that:

$$\tilde{\mathbf{y}}_{(k)} = \mathbf{y}_{(k)}(B) + \mathbf{H}_{(k)}(E) \mathbf{s}_{(k)}, \quad (42)$$

and uses a modified two-loop recursion to provide an improved approximation of the Hessian inverse. Alg. 2 gives detailed pseudocode of our quasi-Newton step. The highlighted texts mark our modifications out of struck-through steps in the standard L-BFGS procedure.

Note that at line 12 of Alg. 2, we need to solve $H_{(k)}(E) + H_{(1)}(B)$ at each iteration e.g., using Cholesky factorization. As the sparsity pattern is unchanged, we reuse the previous symbolic factorization. After the descent direction is computed, we apply an Armijo backtracking line search to update q .

ALGORITHM 2: Modified L-BFGS iteration

Input: $q(k), q(k-1), \{s(1), \dots, s(k-2)\}, g(k),$
 $\{y(1), \dots, y(k-2)\},$
 $\{y(1)(B), \dots, y(k-2)(B)\},$
 $g(k-1),$
 $g(k-1)(B, q),$
 $H_{(1)},$
 $H_{(1)}(B)$
Output: δq

```

1  $d \leftarrow g(k)(E + B, q)$ 
2  $s(k-1) \leftarrow q(k) - q(k-1)$ 
3  $y(k-1) \leftarrow g(k) - g(k-1)$ 
4  $\{ \text{Assemble elasticity Hessian } H_{(k)}(E) \}$ 
5  $y(k-1)(B) \leftarrow g(k)(B, q) - g(k-1)(B, q)$ 
6 for  $i \leftarrow k - m$  to  $k - 1$  do
7    $\tilde{y}_{(i)} \leftarrow y_{(i)}(B) + H_{(k)}(E)s_{(i)}$ 
8 end
9 for  $i \leftarrow k - 1$  to  $k - m$  do
10    $\alpha_{(i)} \leftarrow \frac{s_{(i)}^T d}{\tilde{y}_{(i)}^T s_{(i)}}$ 
11    $d \leftarrow d - \alpha_{(i)} \tilde{y}_{(i)}$ 
12 end
13  $\text{Solve } H_{(1)} \delta q = d \text{ with Eq. (44)}$ 
14  $\{ \text{Solve } (H_{(k)}(E) + H_{(1)}(B)) \delta q = d \text{ exactly} \}$ 
15 for  $i \leftarrow k - m$  to  $k - 1$  do
16    $\beta_{(i)} \leftarrow \frac{\tilde{y}_{(i)}^T \delta q}{\tilde{y}_{(i)}^T s_{(i)}}$ 
17    $\delta q \leftarrow \delta q + (\alpha_{(i)} - \beta_{(i)}) s_{(i)}$ 
18 end
19  $\delta q \leftarrow -\delta q$ 

```

7 EXPERIMENTAL RESULTS

We implemented VABD framework on a desktop workstation equipped with an Intel i7-12700F CPU and an NVIDIA RTX 3080 GPU. Table 1 summarizes the simulation setups of all the experiments reported in the paper, and Table 2 gives the detailed timing statistics. Unless otherwise noted, the simulation pipeline runs on the GPU. We use a St. Venant Kirchhof material model for generating weight modes and precomputing the rest-shape Hessians. For all experiments in this paper, we applied a relative residue tolerance of 1×10^{-3} as termination criteria, with a maximum of 8 iterations per time step. Sparse linear systems are solved using cuDSS at full Newton steps in our NQN procedure. Neural network training is done using CUDA-enabled PyTorch [Paszke et al. 2019]. Given trained network weights, we further implemented a dedicated CUDA inference kernel along

Table 1. Experiment setups. This table reports the configurations used in our scenes. *Freq.* denotes the DCD and CCD frequency, measured in Hz. *#obj.*, *r*, *#ele*, *#pts*, and *#MPs* report the number of objects, the number of affine modes per object, the number of elements and number of nodes in the FE mesh, and the total number of medial primitives, respectively. The last two columns report the Young’s modulus and Poisson’s ratio of the material (E, ν).

Scene	Freq. (Hz)		#obj.	<i>r</i>	#ele.	#pts.	#MPs	<i>E</i> (MPa)	ν
	DCD	CCD							
Fig. 1	500	100	356	10	5.1M	1.7M	75K	5.0	0.25
Fig. 6	500	100	2	10	22.2K	6.5K	30	500	0.25
Fig. 11	200	40	2	10	0.11M	28.5K	200	50	0.25
Fig. 12	1000	200	1	10	14.2K	4.8K	211	25	0.25
Fig. 14	500	100	20	10	0.28M	96K	4.2K	5.0	0.25
Fig. 15 (L)	500	100	1	30	0.40M	0.11M	1.3K	25	0.25
Fig. 15 (R)	500	100	1	30	0.40M	0.11M	1.3K	12.5	0.25
Fig. 16	500	100	1	30	51K	17K	11K	250	0.25
Fig. 17	500	100	61	20	0.92M	0.23M	2.5K	50	0.25
Fig. 18	125	25	18	10	267K	137K	380	100	0.25

with a custom derivative kernel to efficiently evaluate the network outputs at runtime for retrieving the updated medial vertices and radii, as well as their gradients $\frac{\partial p}{\partial q}$ as described in Eq. (52). We set the ABD stiffness $k_A = \frac{1}{4}\mu$, where μ is the Lamé parameter. Comparing the ABD energy $E_A = k_A V \|AA^T - I\|^2$ with the St. Venant–Kirchhoff (St.VK) model $\Psi = \frac{1}{4}\mu \|F^T F - I\|^2 + \frac{\lambda}{8} \text{tr}(F^T F - I)^2$, this choice captures the stretching term $\frac{1}{4}\mu \|F^T F - I\|^2$ while neglecting the volume-preservation term.

7.1 Comparison with Displacement-based Subspace

In the first experiment, we compare our homogeneous and heterogeneous affine subspace with existing displacement-based model reduction techniques, i.e., using modal derivatives [Barbić and James 2005]. A fullspace stiff FEM simulation is used as the ground truth for the reference. The results are shown in Fig. 6. In this comparison, two tuning forks are placed in a crossed configuration. One tuning fork (fork A) is initially oriented approximately upright, with one end in near contact with the ground. The other tuning fork (fork B) is placed flat, above the ground. The angle between the two tuning forks is approximately 90 degrees. Under the effect of gravity, both fork A and fork B fall, slide rightwards, and eventually come to rest on the ground. In this experiment, we used 20 derivative modes including the first 10 unconstrained eigen modes and 10 modal derivatives. VABD has 10 modes with one homogeneous affine modes and 9 heterogeneous affine bases. All modes are constructed at the rest configuration of the fork A.

Technically, this simulation is not particularly challenging. Yet, it shows a representative case highlighting the difference between the deformation subspace (ours) and the displacement subspace (modal derivative). The deformation on tuning forks is small, and therefore, the motions of fork A and fork B simulated using our method closely match the ground-truth results. In contrast, displacement-based modes are not rotation-invariant: they need to capture changes of *displacement* rather than *deformation*. As a result, fork B exhibits

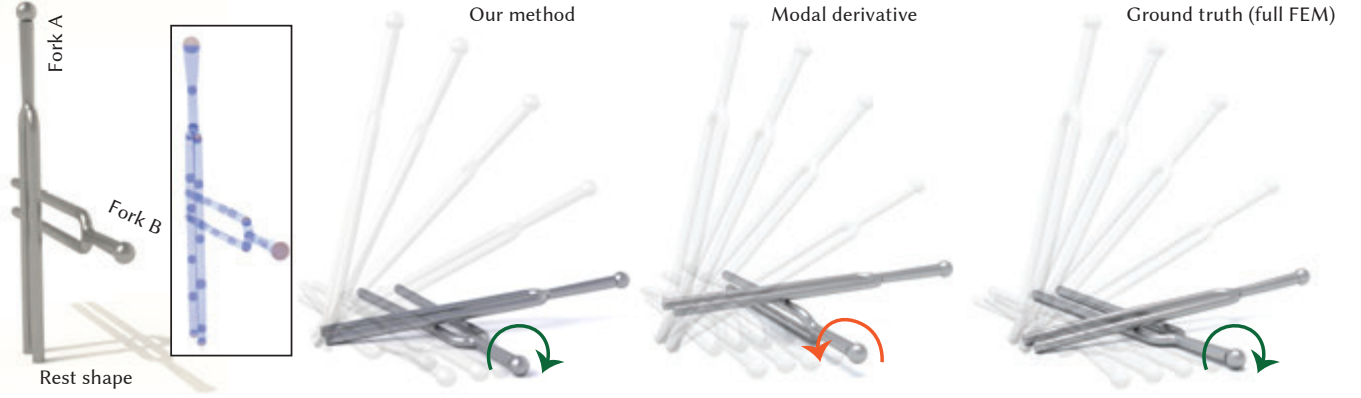


Fig. 6. Comparison with displacement-based model reduction. Displacement modes are frame-dependent. With limited modal bases, it may lead to incorrect behavior when simulating free-floating objects under large rotations. We illustrate this issue of displacement-based model reduction for the tuning fork model (11,082 elements) — the modes are computed when the fork is positioned vertically. They are used for both forks A and B, with fork B being rotated by 90° . We give fork A a small initial velocity to the right so that it collides with fork B so that B also slides towards the right. Under large rotations, modal derivatives (with 20 modes) fail to represent the needed displacement, making fork B turn to the opposite side, although no torque should be driving that action intuitively. On the other hand, our method produces the same motion as the ground truth with the same number of modes. Both reduced simulation methods are over $70\times$ faster than the fullspace FEM. Because of medial CCD/DCD, our method is $30\times$ faster for collision processing.

Table 2. Time statistics. This table lists detailed time statistics of our experiments. *N-MAT* refers to the time used to inference neural MAT, and $\nabla N\text{-MAT}$ is the time for calculating the network gradient/Hessian. $\nabla^2 E$ and $\nabla^2 B$ are computational times for calculating the elasticity and collision Hessian. *Assembly* gives the time for system matrix assembly. *# col* reports an average number of MP-based collisions per time step. *DCD* and *CCD* are the timing for collision detections. *# It.* denotes the average number of iterations per timestep. The *Avg. it.*, *Avg. frame* and *Max. frame* columns report the average iteration time, average frame time, and worst frame time, respectively. All times are reported in milliseconds (ms). We also report the maximum volume change (ΔV) and maximum orthogonality deviation of the affine frame ($\|AA^T - I\|^2$) over the simulation to quantify object deformation.

Scene		N-MAT	$\nabla N\text{-MAT}$	$\nabla^2 E$	$\nabla^2 B$	Assembly	Solve	DCD	CCD	# Col.	# It.	Avg. it.	Avg. frame	Max. frame	FPS	ΔV	$\ AA^T - I\ ^2$
Spiky balls I	Fig. 1	27.4	29.7	2.2	30.6	105.4	280.5	61.3	338.8	4.0K	5.6	876.0	4939.5	19499.9	0.2	19.9%	1.86×10^{-2}
Forks, ours	Fig. 6	0.2	0.2	0.2	1.3	1.0	6.3	0.8	2.1	9.7	2.2	12.1	27.2	139.4	36.7	1.48%	1.84×10^{-6}
Forks, MD	Fig. 6	N/A	N/A	15.5	12.7	12.1	103.2	13.7	82.0	80	3.7	239.1	893.7	5133.4	1.1	0.11%	N/A
Forks, FEM	Fig. 6	N/A	N/A	4.2	18.1	15.8	50.5	9.8	69.5	78	3.9	167.9	647.6	3851.0	1.5	0.11%	N/A
Funnel, ours	Fig. 12	0.2	0.5	0.1	0.7	0.8	1.5	1.1	1.5	3.4	1.9	6.5	12.3	120.9	81.2	10.9%	2.12×10^{-3}
Funnel, ABD	Fig. 12	N/A	N/A	0.1	0.3	0.3	1.2	1.5	3.6	7.1	1.9	7.0	13.5	124.7	74.1	2.28%	4.91×10^{-3}
Spiky balls II, ours	Fig. 14	1.4	2.8	0.4	2.0	12.3	11.9	4.2	5.5	36.4	3.5	40.3	143.0	909.0	7.0	12.8%	1.07×10^{-2}
Spiky balls II, FCD	Fig. 14	1.3	2.7	0.4	1.9	11.7	12.5	4.6	5.8	34.8	3.9	41.0	161.1	1081.7	6.2	9.4%	7.57×10^{-3}
Ship on stairs	Fig. 15	0.2	0.5	0.2	5.3	12.7	5.4	1.0	1.4	4.2	2.0	26.8	53.4	294.9	18.7	3.63%	5.43×10^{-5}
Ship on slope	Fig. 15	0.2	0.7	0.2	3.6	16.2	6.1	1.0	1.4	16.0	1.9	29.4	55.7	277.4	18.0	2.78%	4.91×10^{-3}
Sound synthesis	Fig. 16	0.4	0.9	0.6	6.0	13.4	2.3	1.6	3.8	3.7	1.8	29.1	53.4	221.8	18.7	0.67%	1.51×10^{-4}
Windmill	Fig. 17	4.3	5.4	0.4	2.9	3.4	70.9	3.1	16.6	43.9	6.0	107.1	647.7	1133.8	1.5	6.46%	1.50×10^{-3}
Crane	Fig. 18	0.3	0.4	0.2	3.3	13.5	6.9	1.3	1.8	19.8	2.7	27.8	74.4	237.6	13.4	6.60%	3.58×10^{-3}

qualitatively different behavior and topples toward the left instead. We emphasize that VABD is only an approximation to the full-order simulation governed by the same St.VK material model. Failure modes inherent to linear subspace reduction, such as volume loss under large deformation, may therefore still occur in our method.

In this experiment, there are 11,082 elements on each tuning fork, and we generate a neural MAT consisting of 15 MPs for collision handling. While both our method and modal derivative take similar time for solving the elasticity, our method is more efficient at the collision detection and resolution step, which is $30\times$ faster.

7.2 Comparison with FCD

As discussed in Sec. 4.2, our method is highly relevant to FCD [Benchekroun et al. 2023] as both encode deformation modes through spatially varying weighting fields. Specifically, our method builds subspace bases for desired deformations via Eqs. (18) and (20), whereas FCD requires solving an optimization with a prescribed deformation covariance matrix. The difference between our method and FCD is subtle, yet our method tends to give better reconstruction when a specific set of deformable poses is needed. To illustrate this, we compare the expressivity of our method against FCD for three standard low-frequency deformation poses of a rectangular beam: nonlinear bending, twisting, and shearing. As shown in Fig. 7, both FCD and our method use 8 weighting modes. The figure reports

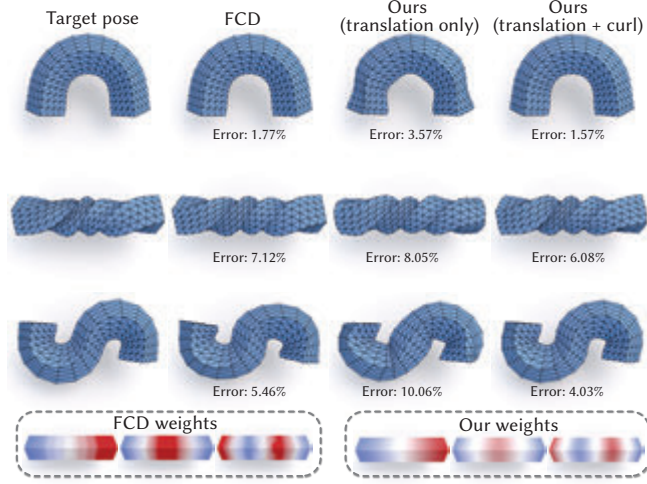


Fig. 7. Comparison with FCD I. The rectangular beam consists of 1,920 tetrahedral elements, and it undergoes three standard nonlinear deformations: bending, twisting, and shearing. We use FCD and our heterogeneous affine subspace to reconstruct those three poses using 8 weighting modes. Both methods manage to represent the target poses with small errors. We also visualize the distributions of the first three weights of both methods.

the best-matching subspace deformation each method can express for a given target pose. Specifically, let $\mathbf{u}^*$ be the deformation of interest, and $\mathbf{U}$ be the basis matrix of the linear subspace (either FCD or ours). The best-matching subspace deformation is computed as: $\mathbf{U}(\mathbf{U}^\top \mathbf{U})^{-1} \mathbf{U}^\top \mathbf{u}^*$. For simple geometries like this beam, our method performs comparably to FCD. It can be seen from the figure that incorporating the curl-based weighting field effectively enhances the expressivity of our subspace.

Fig. 8 plots the evolution of the reconstruction error with increasing number of weight modes, for a spiky ball model with 42 tentacles. The target pose is taken from a reference FEM simulation of the scene in Fig. 12, which includes observable local deformation at the spikes. In this case, both our method and FCD weighting scheme are able to capture these local details using 42 weighting modes. As with Fig. 8, the incorporation of curl-based weights (Eq. (20)) significantly improves the overall reconstruction quality.

Lastly, we directly compare our modes with the FCD modes in the dynamic scene shown in Fig. 14. With FCD weights, the simulation remains stable and free of visible artifacts, while achieving comparable computational performance and strain magnitudes to our method, as summarized in Table 2.

It is never our intention to claim that our method is “better” than FCD in a general setting. In many applications, the reconstruction of specific poses should be prioritized within the dynamic subspace, and these poses may not always correspond to those with smaller energies. In these cases, user-supplied displacement modes are useful for bootstrapping the deformation subspace. Our method is fully compatible with this workflow, as it takes specific target poses as input to extract corresponding rotation-invariant information. In contrast, FCD modes are computed without prior knowledge of the

target pose. Therefore, we consider our proposed heterogeneous affine subspace a useful complement to FCD.

7.3 Neural MAT

This section elaborates on the implementation of the neural MAT. We cover data generation, network layer width, and derivative computations. Finally, we benchmark the neural MAT against Medial IPC [Lan et al. 2021] to demonstrate its effectiveness.

7.3.1 Data generation. For each tetrahedral model used in the simulation, we apply Poisson disk sampling [Bridson 2007] around the rest pose to create 2,000 samples of $\tilde{\mathbf{q}}_V$, with the bounded energy constraint of $\tilde{\mathbf{q}}_V^\top \tilde{\mathbf{H}} \tilde{\mathbf{q}}_V \leq 10^{-3}$.

We restore each valid $\tilde{\mathbf{q}}_V$ to its corresponding affine transformations, and apply the affine weights to deform the mesh. We employ dual quaternion skinning [Kavan et al. 2007] to handle large deformations. With the deformed meshes, we compute the optimal medial sphere centers and radii by minimizing the spherical quadratic error metric (SQEM) [Li et al. 2016; Thierry et al. 2013]. This metric accumulates the area-weighted sum of squared sphere-plane distances. We establish the binding between surface patches and medial primitives in the rest configuration and keep it fixed during deformation. Under this assumption, the mapping from $\tilde{\mathbf{q}}_V$ to the optimal medial state $\mathbf{p}^*$ is expressed as:

$$\mathbf{p}^*(\tilde{\mathbf{q}}_V) = \arg \min_{\mathbf{p}} \text{SQEM}(\mathbf{p}, V(\tilde{\mathbf{q}}_V)). \quad (43)$$

Neural MAT approximates the optimization of Eq. (43) with a lightweight neural network. To train the net, we pre-compute the pairs of generalized coordinates and medial state vectors $(\tilde{\mathbf{q}}_V, \mathbf{p}^*(\tilde{\mathbf{q}}_V))$ across all sampled poses. 75% of the resulting data were used for training, while the remaining 25% was for testing. The training process employs the AdamW optimizer [Loshchilov and Hutter 2017] with a learning rate of 10^{-4} . Fig. 10 shows that the network converges within 6,000 epochs, taking approximately 6 min.

7.3.2 Comparison with medial IPC. MAT deforms the medial mesh based on the generalized coordinates $\tilde{\mathbf{q}}_V$. A simpler alternative would embed each medial sphere into the tetrahedral mesh via barycentric interpolation. This method has been a popular choice in prior works such as medial IPC [Lan et al. 2021]. The centers of medial spheres displace with the embedded tetrahedra, and radii remain unchanged. While easy to implement, the limitations are also obvious. Treating a sphere as a single embedded point negatively impacts the tightness of deformed MAT.

Our neural MAT avoids these limitations. The network predicts both the position and radius of each medial sphere directly from $\tilde{\mathbf{q}}_V$. Fig. 9 compares our method with medial IPC. Our method allows medial spheres to move across tetrahedral boundaries and settle into positions that better capture the surface geometry. Consequently, the predicted spheres well approximate the tangent of the surface, ensuring a high approximation quality. In contrast, medial IPC yields a looser MAT enclosure under large and local deformations. In this experiment, the Eiffel Tower model contains 23,192 vertices and 93,942 elements. We approximate its surface using 150 medial spheres. Our neural MAT computes the deformed medial mesh within 1 ms.

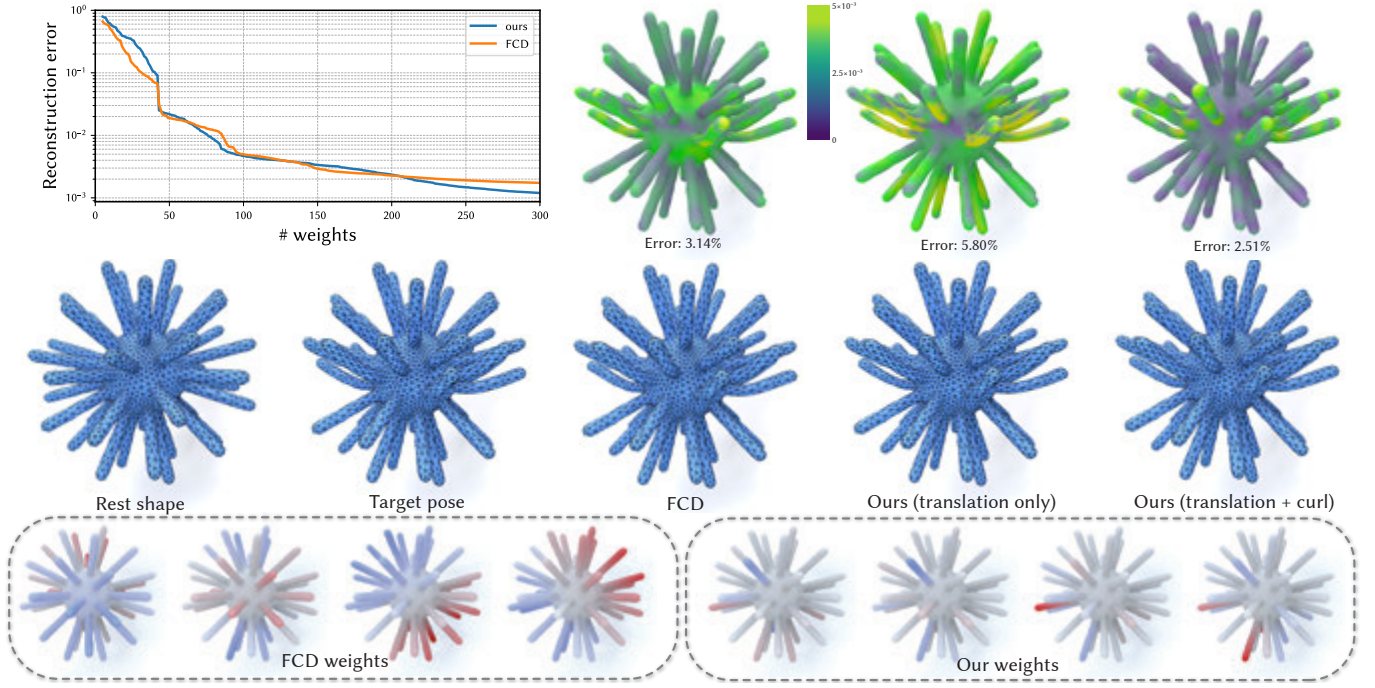


Fig. 8. Comparison with FCD II. Using 42 weighting modes for a 42-tentacle spiky ball, our method achieves reconstruction error comparable to FCD, while providing easier conversion from displacement-based modes. Including curl-based weights further improves the reconstruction error. In this experiment, the target pose is captured from FEM simulation of the scene in Fig. 12.

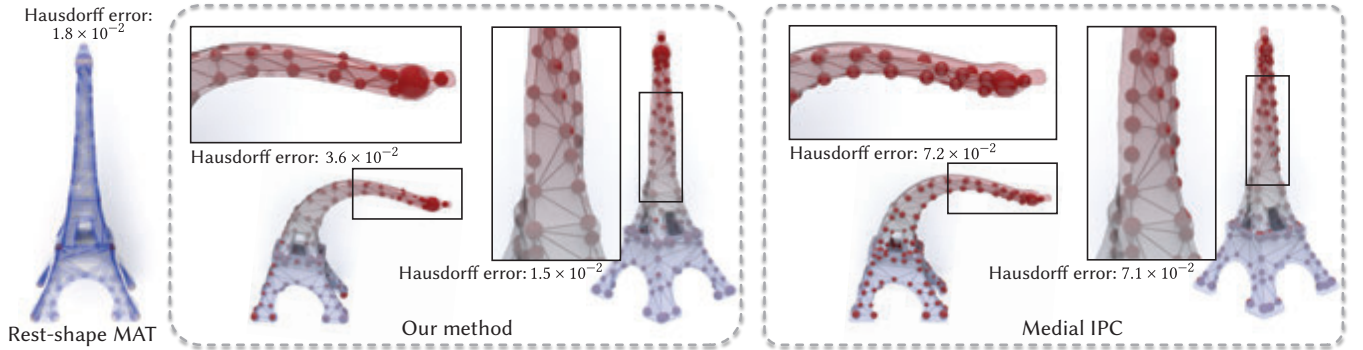


Fig. 9. Comparison with medial IPC. Neural MAT predicts a high-quality MAT for an input reduced deformation. Compared with interpolation-based MAT update e.g., used in medial IPC [Lan et al. 2021], our method keeps a tighter enclosure of the deformed model. In this example, the rest-shape MAT of the Eiffel Tower model has a Hausdorff error of 1.8×10^{-2} . Under nonlinear bending and twisting, medial IPC increases the errors to 7.2×10^{-2} and 7.1×10^{-2} , while neural MAT results in much lower errors: 3.6×10^{-2} and 1.5×10^{-2} .

7.3.3 Network parameter and inference. While $\mathbf{p}$ and $\tilde{\mathbf{q}}_V$ are certainly nonlinearly related, we consider the underlying nonlinearity moderate. Therefore, an MLP-based network should suffice. Several prior works also suggest that MLP becomes less effective as the depth increases [Poole et al. 2016; Schoenholz et al. 2016]. Therefore, we set the depth of neural MAT as in Fig. 4.

We also investigate the impact of the hidden layer width by measuring inference time, gradient computation time, and validation

error using the Eiffel Tower model. The statistic is reported in Fig. 10. The validation loss is based on the L2 norm. While increasing the width leads to smaller fitting errors, both inference time and gradient computation time go up as well. The accuracy gain becomes less profitable after the width reaches 128 i.e., the inference time takes more than a few milliseconds. Based on this empirical observation, we choose to set the width as 64 for all the experiments.

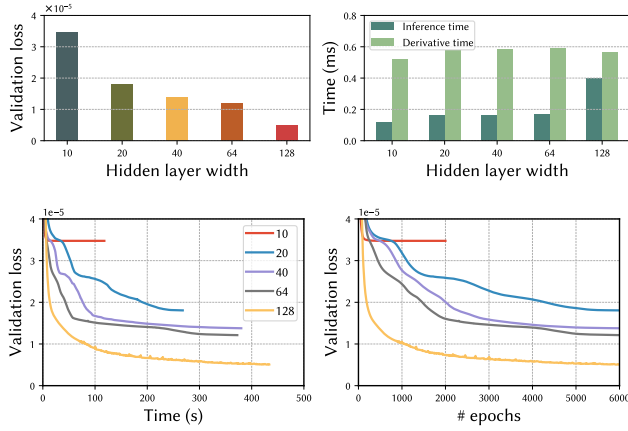


Fig. 10. Network architecture. (Top) We measure the influence of hidden layer width on validation loss, inference time, and derivative time. We found that setting the width to 64 provides a good trade-off between prediction accuracy and efficiency. The total computation time related to neural MAT is less than 1 ms while the prediction remains high-quality i.e., see Fig. 9. (Bottom) The neural network training takes around 6 minutes and 6,000 epochs to converge.

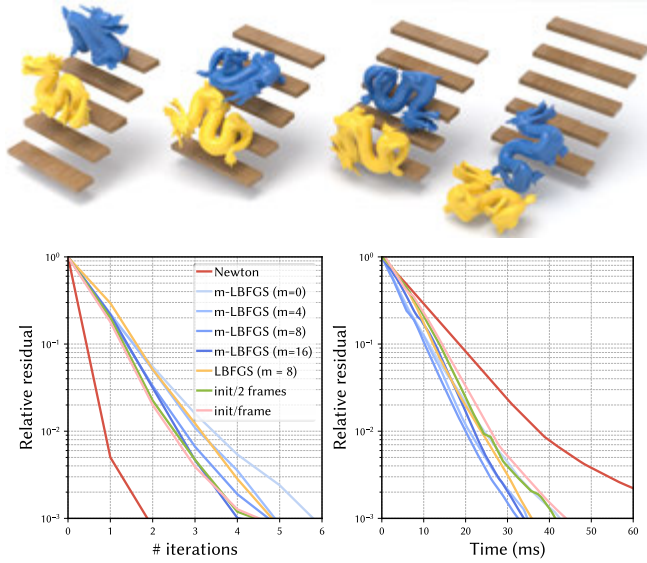


Fig. 11. Convergence plot. We compare the convergence of multiple L-BFGS variants with the Newton iteration in a standard test, where two stiff dragons slide down the stairs (as shown in the top). Our default modified L-BFGS solver, denoted *m-LBFGS* ($m = 8$), stores a history length of $m = 8$ and updates the Hessian every 5 DCD substeps of $h = 1/500$ s. Modified L-BFGS with different history length is tested and labeled as *m-LBFGS* ($m = 0$), *m-LBFGS* ($m = 4$) and *m-LBFGS* ($m = 16$). Standard LBFGS ($m = 8$) is also evaluated as a baseline. For $m = 8$, we also test alternative Hessian-reuse strategies: recomputing once per DCD step (*init/frame*) or once every two steps (*init/2 frames*). Our *m-LBFGS* solver with history length $m = 8$ outperforms the Newton iterations by 90% in computation time.

7.4 Convergence

We compare the performance of our modified L-BFGS solver against the standard L-BFGS solver and baseline Newton solver. The experiment scene and convergence rate are illustrated in Fig. 11. Overall, the modified L-BFGS solver offers a favorable balance between iteration efficiency and computational cost, achieving approximately 90% speedup over Newton iterations, and up to 20% improvement relative to the standard L-BFGS.

Within the modified L-BFGS framework, we further analyze the convergence behavior under different history lengths and Hessian initialization intervals. Our default implementation, denoted *m-LBFGS* ($m = 8$), uses a history size of $m = 8$, and recomputes the exact Hessian every 5 time steps. Comparison of different history lengths is plotted for $m = 0$, $m = 4$, and $m = 16$. For the same history length, we test two Hessian-reuse strategies, assessing the quality of the Hessian approximation when reinitialized once per frame (*init/frame*) and once every two frames (*init/2 frames*). We used a relative error tolerance of 10^{-3} for the experimental settings.

7.5 Analysis of Subspace and Medial Proxy Resolution

Fig. 13 presents an ablation study of the two primary parameters controlling the accuracy–performance tradeoff in our system: the number of deformation weight modes and the number of medial primitives used for collision proxy.

For the study on the subspace size, we measure the average reconstruction error defined in Sec. 7.2 over the reference FEM simulation sequence from Fig. 11. The number of medial primitives is fixed to 100, and all reduced systems are solved using the Eigen dense LDLT solver. Increasing the number of weight modes consistently improves reconstruction fidelity; however, the cost of Hessian assembly and dense linear solves grows super-linearly with the subspace dimension. In our experiment, increasing the number of modes from 40 to 160 reduces the reconstruction error by approximately 50%, while increasing the total computation time from 39 ms to 977 ms per step.

For the collision proxy study, we vary the number of medial primitives while fixing the number of weight modes to 10. We measure the approximation quality using the Hausdorff distance between the medial proxy and the rest-shape surface mesh. Although our implementation does not employ broad-phase collision culling, collision detection remains a minor portion of the total runtime; instead, collision Hessian assembly dominates the per-iteration cost. In this setting, the total simulation time scales approximately linearly with the number of medial primitives.

7.6 Comparison with ABD

The heterogeneous affine subspace enriches affine bodies with subtle but vivid dynamics. For example, in Fig. 12, we drop a stiff spiky ball through a funnel that is slightly narrower than the ball’s size. Under standard affine body settings, the ball is blocked at the funnel opening, no matter the weight of the ball. In contrast, when 10 vibrational DOFs are introduced, the spikes become enlivened and bend in compliance with the funnel shape, eventually passing through the gap. The ABD simulator simulates a triangle mesh with 8,120 triangles and 4,062 vertices. The Young’s modulus and Poisson’s ratio E, ν

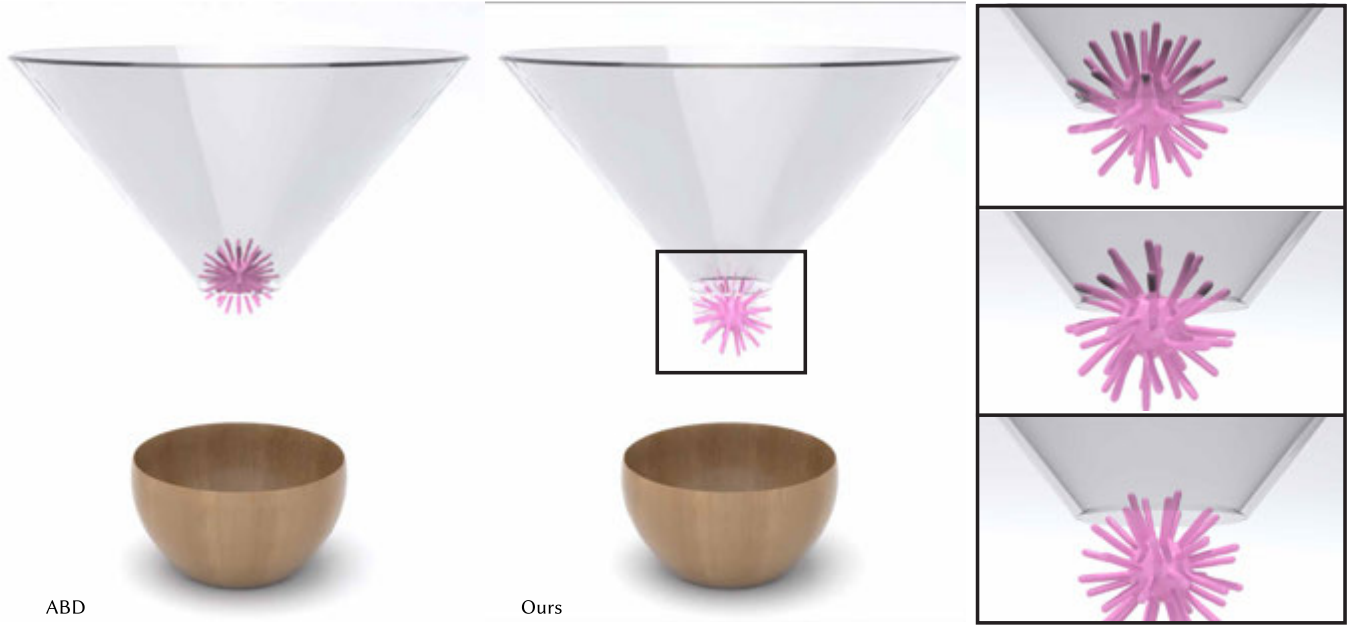


Fig. 12. Comparison with ABD. A spiky ball falls through the funnel whose diameter is slightly smaller than the size of the ball. With affine motions only, the ball barely deforms and gets stuck above the opening. Our augmented vibration modes help spikes on the ball to deform accordingly while keeping its tentacles in contact with the funnel, and eventually drop through the funnel.

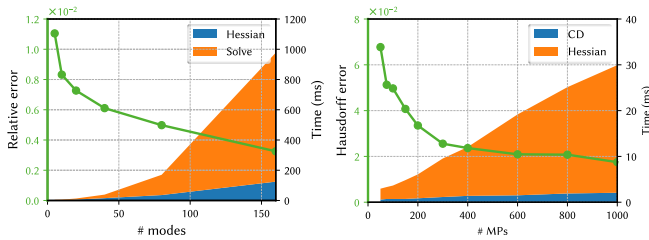


Fig. 13. Ablation study on subspace size and collision proxy resolution. We analyze the tradeoff between simulation accuracy and computational cost using the example from Fig. 11. (Left) Effect of the number of weight modes on reconstruction accuracy and simulation time, with the number of medial primitives fixed to 100. Increasing the subspace dimension improves reconstruction quality but leads to a super-linear increase in solver cost. (Right) Effect of the number of medial primitives on collision proxy approximation error and simulation cost, with the number of weight modes fixed to 10. Increasing proxy resolution improves geometric fidelity while producing an approximately linear increase in runtime.

are set to 25 MPa and 0.25 respectively. In comparison, our VABD solver sees the spiky ball as a medial mesh with 211 MPs. While our system size is 10× larger than ABD, our NQN solver only uses 25% more time at each frame, and achieves real-time performance (over 60 FPS).

The accompanying video also includes more side-by-side comparisons with ABD for the scenes shown in Figs. 14 and 15. The

added dynamic richness of VABD is clearly observable. In the experiment shown in Fig. 14, twenty squishy balls are released and fall between two walls. We also set multiple rows of wooden rods in the space. As the balls fall, they exhibit intricate contact interactions both with the rods and among themselves. In this simulation, each spiky ball is augmented with 10 affine weights, and the simulation runs at 8 FPS. If treated as a simple affine body, each ball behaves like a rigid sphere, and the spikes on the ball appear over-stiffened. However, VABD successfully captures the intended dynamics of the spikes. Compared with reduced-order stiff FEM simulations, VABD utilizes significantly fewer modes and achieves orders-of-magnitude speedups in collision processing.

In Fig. 15 (left), the barbarian ship model slides down a spiral stair. During its descent, the thin structures respond and vibrate upon impact with the stairs. On the right, the same ship is dropped on a slope and lands directly on its paddles. The paddles, due to their thin geometries, exhibit interesting deformations, bending backward to absorb the impact energy. In contrast, ABD models the ship as a largely homogeneous body, resulting in a boring sliding motion that fails to reflect geometry-specific deformations. In this experiment, the ship’s stiffness is set to 12.5 MPa with a time step of $h = 1/500$ s. The ABD simulator operates on a triangular collision mesh with 148K triangles and 79K nodes, whereas our VABD simulator represents the geometry with only 1.3K medial primitives.

7.7 Sound Synthesis

A natural advantage of VABD is that the captured high-frequency deformations not only enhance visual fidelity, but also improve the realism of sound synthesis. In acoustic simulation, high-frequency



Fig. 14. Spiky balls. Twenty spiky balls are released into a space interspersed with wooden rods, exhibiting intricate body-body contact. Compared to ABD, where the spikes appear over-stiffened and the ball behaves like a rigid sphere, our VABD successfully captures the flexible dynamics of the spikes using only 10 affine weights.

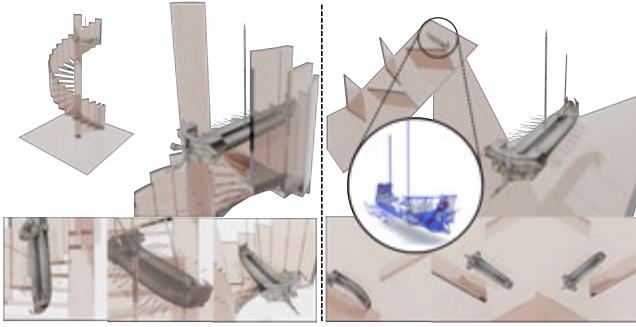


Fig. 15. Barbarian ship. (Left) The ship slides down a spiral stair, where VABD captures the characteristic vibrations of thin structures induced by impact. (Right) When dropped on a slope, the paddles exhibit realistic bending and energy absorption. While standard ABD results in a single sliding motion due to its homogeneous body assumption, VABD successfully captures detailed geometry-specific deformations.



Fig. 16. Sound synthesis. Compared to standard ABD, our VABD framework leverages orthogonal vibrational modes to capture high-frequency surface oscillations upon impact. This additional vibrational information is essential for synthesizing realistic sound, which cannot be achieved by simply increasing the temporal sampling rate of rigid or soft-body solvers.

sampling of surface vibrations is essential. Without vibrational motion, simply increasing the temporal sampling rate (i.e., reducing

the time step size) provides limited improvement to sound quality. However, VABD leverages the MAT and heterogeneous affine displacements orthogonal to standard ABD motion to efficiently compute nonlinear high-frequency vibrations induced by collisions. This vibrational information effectively enhances the quality and authenticity of the synthesized sound. Fig. 16 reports such an example. In this experiment, we select a complex 3D model from the KleinPAT dataset [Wang and James 2019], which has multiple layered, radially oriented fins. When this model impacts the floor, we synthesize the sound produced by the impact using both ABD and VABD. As expected, the high-frequency vibrations simulated with VABD significantly enrich the resulting acoustic signals, leading to a more natural and realistic sound.

7.8 More Examples

We show more examples in Figs. 17, 1, and Fig. 18. In Fig. 17, 60 bunnies are dropped onto a windmill. Collisions transfer momentum to the wheel, triggering its rotation around the pinned axis. Repeated impacts cause deformations in the windmill blades to accumulate, which in turn pushes the bunnies in multiple directions. The rebounding bunnies exhibit rich secondary vibrations, particularly in their ears, highlighting the expressiveness enabled by our additional DOFs. In this simulation, each bunny is assigned 20 weight modes and utilizes a medial mesh with 40 MPs as a collision proxy. Similarly, the windmill features 20 weight modes with the same material parameters, and its medial mesh consists of 100 MPs (as highlighted in Fig. 17). The windmill is hinged by two linear equality constraints that fix the positions of two points along its axis. These constraints are efficiently handled via a change-of-variables strategy following [Chen et al. 2022].

Our robust collision resolution is best demonstrated in the teaser scene (Fig. 1), which simulates 356 spiky balls being released into a stack of bowls. The system exhibits complex cascading dynamics: as the upper bowls reach their capacity, the balls overflow and populate

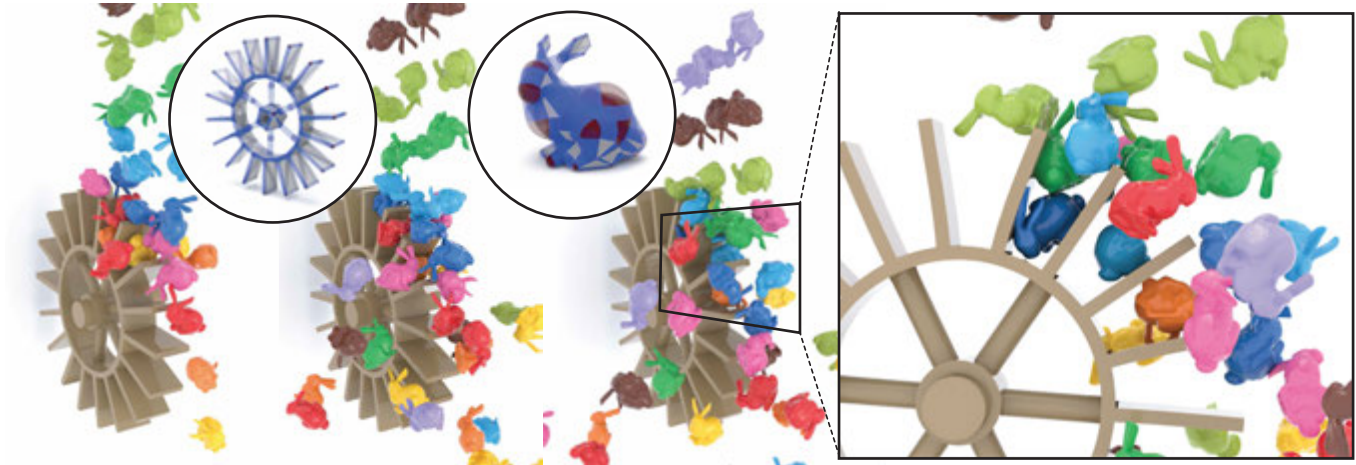


Fig. 17. Falling bunnies on the windmill. This scene contains 60 falling bunnies and a windmill, both simulated with the VABD simulator. We augment the affine-body motion with 20 vibration modes to both the bunnies and the windmill, resulting in richer dynamics. Impacts from the bunnies not only induce rotational motion of the windmill but also cause its blades to deform and vibrate. Collision detection and handling in this experiment are executed on medial primitives. The windmill is converted to a medial mesh with 100 MPs, and the bunnies are converted to medial meshes with 40 MPs.



Fig. 18. Crane and chain. This experiment highlights the differences in energy storage and transfer between ABD and VABD. VABD’s vibrational DOFs enable the system to efficiently convert gravitational potential energy into elastic energy, capturing both the natural bouncing of the ball and the subtle deformations of the crane arm. In contrast, standard ABD exhibits excessive damping, leading to a stiff and less realistic dynamic response.

the lower tiers level by level. This process triggers a massive number of ball-ball and ball-structure contacts, inducing high-frequency vibrational modes that manifest as energetic bouncing and localized “wagging” of the spikes. While a standard tetrahedral FEM discretization of this domain would involve a prohibitive 5.06M elements and 1.7M vertices, our VABD framework leverages the combination of modal reduction and medial proxies. This approach collapses the system to a mere 42,720 DOFs and 75,116 MPs. Despite

the extreme contact density and geometric complexity, our solver maintains high efficiency, requiring an average of only 4.6 seconds per frame ($h = 1/500$ s) on the GPU.

Our final experiment compares how VABD and ABD handle energy storage and transform. We simulate a crane suspending a heavy ball with a chain made of multiple VABD-based links. When the ball is dropped, a large amount of gravitational potential energy must be converted into elastic energy in a very short time and then transferred back to the ball. VABD’s vibrational DOFs effectively facilitate this energy conversion. Thanks to the increased DOFs, we can clearly observe the deflection of the crane arm caused by the ball’s drop. Under the same simulation settings, however, ABD exhibits much more severe damping. The motion appears stiff and does not match real-world behavior, as it fails to capture the necessary high-frequency response.

8 CONCLUSION

This paper proposes an augmentation of ABD that allows stiff models to exhibit vibrational deformations under high-speed collisions and sharp external forces. From the dynamics modeling perspective, we design a so-called heterogeneous affine subspace on top of a homogeneous affine subspace to capture localized deformation. The subspace is formed as spatially varying weighting fields so that the global affine transformation can be applied to the generalized coordinates. We show that it is possible to build the heterogeneous affine subspace by extracting rotation-invariant features from target poses conveniently. Capturing high-frequency vibration also requires dense time sampling and small time steps. This requirement brings significant computational overhead for collision processing. We leverage the fact that stiff models normally have moderate deformations, and propose a MAT-based collision handling algorithm. This method mixes medial CCD and DCD to balance the efficiency, visual plausibility, and accuracy. To this end, we build a learning-based MAT updating pipeline named neural MAT to predict a tight

MAT enclosure when the model is deformed within the subspace. At the solver level, we introduce a hybrid nonlinear procedure interleaving Newton and quasi-Newton iterations. Those contributions eventually facilitate an efficient and scalable simulation pipeline for vibrational affine bodies.

Our method also has some limitations. While MAT-based collision detection is more efficient than triangle-based solutions, we check MP pairs in a brute-force manner. The quadratic complexity could quickly become noticeable if we further scale up the simulation. For instance, in the example reported in the teaser figure (Fig. 1), MAT-based collision processing uses considerable time. In the future, we will combine the MAT-based method with other broader phase culling strategies, like spatial hashing, to further reduce the size of the MP pairs. Another limitation is the nonlinear solve. While the proposed NQN method is efficient, we note that its convergence may be further improved if well-designed warm-start or preconditioning methods could be used. Given the fact that affine bodies are coupled by contacts, a block-Jacobi-like preconditioner seems to be a good candidate. It is also of strong interest for us to generalize VABD for multibody and articulated bodies. After all, purely rigid objects do not exist in physical reality, and vibrations are inevitable under abrupt external loads. Those subtle deformations may be the key to closing the sim2real gap.

REFERENCES

- Mihai Anitescu and Florian A Potra. 1997. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics* 14, 3 (1997), 231–247.
- David Baraff. 1989. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*. 223–232.
- David Baraff. 1991. Coping with friction for non-penetrating rigid body simulation. *ACM SIGGRAPH computer graphics* 25, 4 (1991), 31–41.
- David Baraff. 1994. Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. 23–34.
- David Baraff. 1997. An introduction to physically based modeling: rigid body simulation 1—unconstrained rigid body dynamics. *SIGGRAPH course notes* 82 (1997).
- Jernej Barbic and Doug L. James. 2005. Real-Time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Trans. Graph.* 24, 3 (July 2005), 982–990.
- Joachim Baumgarte. 1972. Stabilization of constraints and integrals of motion in dynamical systems. *Computer methods in applied mechanics and engineering* 1, 1 (1972), 1–16.
- Otman Benckroun, Jiayi Eris Zhang, Siddhartha Chaudhuri, Eitan Grinspun, Yi Zhou, and Alec Jacobson. 2023. Fast Complementary Dynamics via Skinning Eigenmodes. *ACM Trans. Graph.* 42, 4, Article 106 (July 2023), 21 pages.
- Jan Bender, Kenny Erleben, and Jeff Trinkle. 2014. Interactive Simulation of Rigid Body Dynamics in Computer Graphics. *Comput. Graph. Forum* 33, 1 (Feb. 2014), 246–270.
- Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective dynamics: fusing constraint projections for fast simulation. *ACM Trans. Graph.* 33, 4, Article 154 (July 2014), 11 pages. <https://doi.org/10.1145/2601097.2601116>
- Robert Bridson. 2007. Fast Poisson disk sampling in arbitrary dimensions. *SIGGRAPH sketches* 10, 1 (2007), 1.
- Yue Chang, Otman Benckroun, Maurizio M. Chiaramonte, Peter Yichen Chen, and Eitan Grinspun. 2025. Shape Space Spectra. *ACM Transactions on Graphics* (2025).
- Yue Chang, Peter Yichen Chen, Zhecheng Wang, Maurizio M. Chiaramonte, Kevin Carlberg, and Eitan Grinspun. 2023. LiCROM: Linear-Subspace Continuous Reduced Order Modeling with Neural Fields. In *SIGGRAPH Asia 2023 Conference Papers* (Sydney, NSW, Australia) (SA '23). Association for Computing Machinery, New York, NY, USA, Article 111, 12 pages. <https://doi.org/10.1145/3610548.3618158>
- He Chen, Elie Diaz, and Cem Yuksel. 2023a. Shortest path to boundary for self-intersecting meshes. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–15.
- Peter Yichen Chen, Jinxu Xiang, Dong Heon Cho, Yue Chang, G A Pershing, Henrique Teles Maia, Maurizio M Chiaramonte, Kevin Thomas Carlberg, and Eitan Grinspun. 2023b. CROM: Continuous Reduced-Order Modeling of PDEs Using Implicit Neural Representations. In *The Eleventh International Conference on Learning Representations*.
- Yunuo Chen, Minchen Li, Lei Lan, Hao Su, Yin Yang, and Chenfanfu Jiang. 2022. A unified newton barrier method for multibody dynamics. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–14.
- Hyeong In Choi, Sung Woo Choi, and Hwan Pyo Moon. 1997. Mathematical theory of medial axis transform. *pacific journal of mathematics* 181, 1 (1997), 57–88.
- Michael B Cline and Dinesh K Pai. 2003. Post-stabilization for rigid body simulation with contact and constraints. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, Vol. 3. IEEE, 3744–3751.
- E. Coevoet, S. Andrews, D. Relles, and P.G. Kry. 2020. Distant Collision Response in Rigid Body Simulations. *Computer Graphics Forum* 39, 8 (2020), 113–122. <https://doi.org/10.1111/cgf.14106>
- Erwin Coumans. 2015. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*. 1.
- Evan Drumwright. 2007. A fast and stable penalty method for rigid body simulation. *IEEE transactions on visualization and computer graphics* 14, 1 (2007), 231–240.
- Kenny Erleben. 2007. Velocity-based shock propagation for multibody dynamics animation. *ACM Transactions on Graphics (TOG)* 26, 2 (2007), 12–es.
- Yutao Feng, Xiang Feng, Yintong Shang, Ying Jiang, Chang Yu, Zeshun Zong, Tianjia Shao, Hongzhi Wu, Kun Zhou, Chenfanfu Jiang, et al. 2024. Gaussian splashing: Dynamic fluid synthesis with gaussian splatting. *CoRR* (2024).
- Zachary Ferguson, Minchen Li, Teseo Schneider, Francisca Gil-Ureta, Timothy Langlois, Chenfanfu Jiang, Denis Zorin, Danny M. Kaufman, and Daniele Panozzo. 2021. Intersection-free rigid body dynamics. *ACM Trans. Graph.* 40, 4, Article 183 (July 2021), 16 pages.
- Lawson Fulton, Vismay Modi, David Duvenaud, David IW Levin, and Alec Jacobson. 2019. Latent-space dynamics for reduced deformable simulation. In *Computer graphics forum*, Vol. 38. Wiley Online Library, 379–391.
- Theodore F. Gast, Craig Schroeder, Alexey Stomakhin, Chenfanfu Jiang, and Joseph M. Teran. 2015. Optimization Integrator for Large Time Steps. *IEEE Transactions on Visualization and Computer Graphics* 21, 10 (2015), 1103–1115. <https://doi.org/10.1109/TVCG.2015.2459687>
- Dewen Guo, Zhenhong Wang, Minchen Li, Sheng Li, Guoping Wang, Huamin Wang, Chenfanfu Jiang, and Yin Yang. 2026. Heterogeneous Subspace Corrections for GPU Deformable Multibody Dynamics. *ACM Trans. Graph.* 45, 4, Article 100 (July 2026), 16 pages. <https://doi.org/10.1145/3811275>
- Minghao Guo, Bohan Wang, and Wojciech Matusik. 2024. Medial Skeletal Diagram: A Generalized Medial Axis Approach for Compact 3D Shape Representation. *ACM Trans. Graph.* 43, 6, Article 217 (Nov. 2024), 23 pages.
- Shoichi Hasegawa, Nobuaki Fujii, Katsuhito Akahane, Yasuharu Koike, and Makoto Sato. 2004. Real-time rigid body simulation for haptic interactions based on contact volume of polygonal objects. *Transactions of the Society of Instrument and Control Engineers* 40, 2 (2004), 122–131.
- Zhiyong He, Dewen Guo, Minghao Guo, Yili Zhao, Wojciech Matusik, Hao Su, Chenfanfu Jiang, Peter Yichen Chen, and Yin Yang. 2026. M-ABD: Scalable, Efficient, and Robust Multi-Affine-Body Dynamics. *ACM Trans. Graph.* 45, 4, Article 99 (July 2026), 19 pages. <https://doi.org/10.1145/3811276>
- Jin Huang, Yiyang Tong, Kun Zhou, Huijun Bao, and Mathieu Desbrun. 2011. Interactive Shape Interpolation through Controllable Dynamic Deformation. *IEEE Transactions on Visualization and Computer Graphics* 17, 7 (July 2011), 983–992. <https://doi.org/10.1109/TVCG.2010.109>
- Sergio R. Idelsohn and Alberto Cardona. 1985. A load-dependent basis for reduced nonlinear structural dynamics. *Computers & Structures* 20, 1 (1985), 203–210. [https://doi.org/10.1016/0045-7949\(85\)90069-0](https://doi.org/10.1016/0045-7949(85)90069-0) Special Issue: Advances and Trends in Structures and Dynamics.
- Doug L. James and Dinesh K. Pai. 2002. DyRT: dynamic response textures for real time deformation simulation with graphics hardware. *ACM Trans. Graph.* 21, 3 (July 2002), 582–585. <https://doi.org/10.1145/566654.566621>
- Doug L. James and Dinesh K. Pai. 2004. BD-tree: output-sensitive collision detection for reduced deformable models. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 393–398.
- Danny M Kaufman, Timothy Edmunds, and Dinesh K Pai. 2005. Fast frictional dynamics for rigid bodies. In *ACM SIGGRAPH 2005 Papers*. 946–956.
- Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2007. Skinning with dual quaternions. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games (Seattle, Washington) (I3D '07)*. Association for Computing Machinery, New York, NY, USA, 39–46. <https://doi.org/10.1145/1230100.1230107>
- Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 42, 4, Article 139 (July 2023), 14 pages.
- Lei Lan, Danny M Kaufman, Minchen Li, Chenfanfu Jiang, and Yin Yang. 2022. Affine body dynamics: fast, stable and intersection-free simulation of stiff materials. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–14.
- Lei Lan, Ran Luo, Marco Fratarcangeli, Weiwei Xu, Huamin Wang, Xiaohu Guo, Junfeng Yao, and Yin Yang. 2020. Medial elastics: Efficient and collision-ready deformation via medial axis transform. *ACM Transactions on Graphics (TOG)* 39, 3 (2020), 1–17.
- Lei Lan, Yin Yang, Danny Kaufman, Junfeng Yao, Minchen Li, and Chenfanfu Jiang. 2021. Medial IPC: accelerated incremental potential contact with medial elastics.

- ACM Trans. Graph. 40, 4, Article 158 (July 2021), 16 pages.
- Timothy R. Langlois, Steven S. An, Kelvin K. Jin, and Doug L. James. 2014. Eigenmode compression for modal sound models. *ACM Trans. Graph.* 33, 4, Article 40 (July 2014), 9 pages. <https://doi.org/10.1145/2601097.2601177>
- Minchen Li, Zachary Fergusson, Teseo Schneider, Timothy R Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M Kaufman. 2020. Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. *ACM Trans. Graph.* 39, 4 (2020), 49.
- 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 Transactions on Graphics (TOG)* 35, 1 (2015), 1–16.
- Pan Li, Bin Wang, Feng Sun, Xiaohu Guo, Caiming Zhang, and Wenping Wang. 2016. Q-MAT: Computing Medial Axis Transform By Quadratic Error Minimization. *ACM Trans. Graph.* 35, 1, Article 8 (Dec. 2016), 16 pages.
- Siwang Li, Jin Huang, Fernando de Goes, Xiaogang Jin, Hujun Bao, and Mathieu Desbrun. 2014. Space-time editing of elastic motion through material optimization and reduction. *ACM Trans. Graph.* 33, 4, Article 108 (July 2014), 10 pages. <https://doi.org/10.1145/2601097.2601217>
- Hsueh-Ti Derek Liu, Francis Williams, Alec Jacobson, Sanja Fidler, and Or Litany. 2022. Learning Smooth Neural Functions via Lipschitz Regularization. <https://doi.org/10.48550/ARXIV.2202.08345>
- Stephanie Wenxin Liu, Michael Fischer, Paul D. Yoo, and Tobias Ritschel. 2024. Neural Bounding. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) (SIGGRAPH '24). Association for Computing Machinery, New York, NY, USA, Article 98, 10 pages. <https://doi.org/10.1145/3641519.3657442>
- Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).
- Miles Macklin, Matthias Müller, and Nuttapon Chentanez. 2016. XPBD: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (Burlingame, California) (MIG '16). Association for Computing Machinery, New York, NY, USA, 49–54.
- Daniel Meister, Shinji Ogaki, Carsten Benthin, Michael J Doyle, Michael Guthe, and Jiri Bittner. 2021. A survey on bounding volume hierarchies for ray tracing. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 683–712.
- Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2021. NeRF: representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (Dec. 2021), 99–106.
- Vismay Modi, Nicholas Sharp, Or Perel, Shinjiro Sueda, and David I. W. Levin. 2024. Simplicitis: Mesh-Free, Geometry-Agnostic Elastic Simulation. *ACM Trans. Graph.* 43, 4, Article 117 (July 2024), 11 pages. <https://doi.org/10.1145/3658184>
- Jean J Moreau. 1988. Unilateral contact and dry friction in finite freedom dynamics. In *Nonsmooth mechanics and Applications*. Springer, 1–82.
- Matthias Müller and Nuttapon Chentanez. 2011. Solid simulation with oriented particles. *ACM Trans. Graph.* 30, 4, Article 92 (July 2011), 10 pages.
- Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. 2002. Stable real-time deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Antonio, Texas) (SCA '02). Association for Computing Machinery, New York, NY, USA, 49–54. <https://doi.org/10.1145/545261.545269>
- Jorge Nocedal and Stephen J Wright. 2006. *Numerical optimization*. Springer.
- Zherong Pan, Hujun Bao, and Jin Huang. 2015. Subspace dynamic simulation using rotation-strain coordinates. *ACM Trans. Graph.* 34, 6, Article 242 (Nov. 2015), 12 pages. <https://doi.org/10.1145/2816795.2818090>
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).
- Alex Pentland and John Williams. 1989a. Good vibrations: Modal dynamics for graphics and animation. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*. 215–222.
- A. Pentland and J. Williams. 1989b. Good vibrations: modal dynamics for graphics and animation. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '89)*. Association for Computing Machinery, New York, NY, USA, 215–222. <https://doi.org/10.1145/74333.74355>
- Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. 2016. Exponential expressivity in deep neural networks through transient chaos. *Advances in neural information processing systems* 29 (2016).
- Stéphane Redon, Abderrahmane Kheddar, and Sabine Coquillart. 2002. Fast Continuous Collision Detection between Rigid Bodies. *Computer Graphics Forum* 21, 3 (2002), 279–287.
- Leonardo Sacchi, Etienne Vouga, and Alec Jacobson. 2015. Nested cages. *ACM Trans. Graph.* 34, 6, Article 170 (Nov. 2015), 14 pages.
- Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. 2016. Deep information propagation. *arXiv preprint arXiv:1611.01232* (2016).
- Claude E Shannon. 2006. Communication in the presence of noise. *Proceedings of the IRE* 37, 1 (2006), 10–21.
- Nicholas Sharp, Cristian Romero, Alec Jacobson, Etienne Vouga, Paul Kry, David IW Levin, and Justin Solomon. 2023. Data-free learning of reduced-order kinematics. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–9.
- Siyan Shen, Yang Yin, Tianjia Shao, He Wang, Chenfanfu Jiang, Lei Lan, and Kun Zhou. 2021. High-order differentiable autoencoder for nonlinear model reduction. *arXiv preprint arXiv:2102.11026* (2021).
- David E Stewart. 2000. Rigid-body dynamics with friction and impact. *SIAM review* 42, 1 (2000), 3–39.
- Feng Sun, Yi-King Choi, Yizhou Yu, and Wenping Wang. 2015. Medial meshes—a compact and accurate representation of medial axis transform. *IEEE transactions on visualization and computer graphics* 22, 3 (2015), 1278–1290.
- Roger Tam and Wolfgang Heidrich. 2003. Shape Simplification Based on the Medial Axis Transform. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03) (VIS '03)*. IEEE Computer Society, USA, 63.
- Jean-Marc Thiery, Émilie Guy, and Tamy Boubekeur. 2013. Sphere-Meshes: shape approximation using spherical quadric error metrics. *ACM Trans. Graph.* 32, 6, Article 178 (Nov. 2013), 12 pages.
- Jeffrey C Trinkle, JA Tzitzouris, and Jong-Shi Pang. 2001. Dynamic multi-rigid-body systems with concurrent distributed contacts. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 359, 1789 (2001), 2575–2593.
- Ty Trusty, Otman Benckekroun, Eitan Grinspun, Danny M. Kaufman, and David IW Levin. 2023. Subspace Mixed Finite Elements for Real-Time Heterogeneous Elastodynamics. In *SIGGRAPH Asia 2023 Conference Papers* (Sydney, NSW, Australia) (SA '23). Association for Computing Machinery, New York, NY, USA, Article 112, 10 pages. <https://doi.org/10.1145/3610548.3618220>
- Christoph von Tycowicz, Christian Schulz, Hans-Peter Seidel, and Klaus Hildebrandt. 2013. An efficient construction of reduced deformable objects. 32, 6, Article 213 (Nov. 2013), 10 pages. <https://doi.org/10.1145/2508363.2508392>
- Jui-Hsien Wang and Doug L. James. 2019. KleinPAT: Optimal Mode Conflation for Time-domain Precomputation of Acoustic Transfer. *ACM Trans. Graph.* 38, 4, Article 122 (July 2019), 12 pages. <https://doi.org/10.1145/3306346.3322976>
- Jui-Hsien Wang, Ante Qu, Timothy R. Langlois, and Doug L. James. 2018. Toward wave-based sound synthesis for computer animation. *ACM Trans. Graph.* 37, 4, Article 109 (July 2018), 16 pages. <https://doi.org/10.1145/3197517.3201318>
- Ningna Wang, Hui Huang, Shibo Song, Bin Wang, Wenping Wang, and Xiaohu Guo. 2024. MATTopo: Topology-preserving Medial Axis Transform with Restricted Power Diagram. *ACM Trans. Graph.* 43, 6, Article 218 (Nov. 2024), 16 pages.
- Ningna Wang, Bin Wang, Wenping Wang, and Xiaohu Guo. 2022. Computing Medial Axis Transform with Feature Preservation via Restricted Power Diagram. *ACM Trans. Graph.* 41, 6, Article 188 (Nov. 2022), 18 pages.
- Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. 2023. PhysGaussian: Physics-Integrated 3D Gaussians for Generative Dynamics. *arXiv preprint arXiv:2311.12198* (2023).
- Ryan S. Zesch, Vismay Modi, Shinjiro Sueda, and David IW. Levin. 2023. Neural Collision Fields for Triangle Primitives. In *SIGGRAPH Asia 2023 Conference Papers* (Sydney, NSW, Australia) (SA '23). Association for Computing Machinery, New York, NY, USA, Article 76, 10 pages. <https://doi.org/10.1145/3610548.3618225>
- Jiayi Eris Zhang, Doug L. James, and Danny M. Kaufman. 2024. Progressive Dynamics for Cloth and Shell Animation. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–18.
- Changxi Zheng and Doug L. James. 2011. Toward high-quality modal contact sound. *ACM Trans. Graph.* 30, 4, Article 38 (July 2011), 12 pages. <https://doi.org/10.1145/2010324.1964933>
- Changxi Zheng and Doug L. James. 2012. Energy-based self-collision culling for arbitrary mesh deformations. *ACM Trans. Graph.* 31, 4, Article 98 (July 2012), 12 pages.

A BARRIER GRADIENT & HESSIAN

Implicit time integration requires computing the gradient and the Hessian of the collision barrier functions B_{ccd} and B_{dcd} w.r.t. generalized coordinates:

$$\frac{\partial B}{\partial \mathbf{q}} = \frac{\partial B}{\partial \mathbf{d}} \frac{\partial \mathbf{d}}{\partial \mathbf{q}}, \quad \frac{\partial^2 B}{\partial \mathbf{q}^2} = \frac{\partial^2 B}{\partial \mathbf{d}^2} \left(\frac{\partial \mathbf{d}}{\partial \mathbf{q}} \frac{\partial \mathbf{d}}{\partial \mathbf{q}}^\top \right) + \frac{\partial B}{\partial \mathbf{d}} \frac{\partial^2 \mathbf{d}}{\partial \mathbf{q}^2}. \quad (44)$$

They are evaluated through the dependence chain of: $B \rightarrow d \rightarrow \mathbf{p} \rightarrow \mathbf{q}$. The first- and second-derivative of B_{ccd} w.r.t. medial distance (when $d > 0$) are:

$$\begin{aligned} \frac{\partial B_{ccd}}{\partial d} &= -\kappa_{ccd} \left[2(d - \hat{d}) \ln \left(\frac{d}{\hat{d}} \right) + \frac{(d - \hat{d})^2}{d} \right], \\ \frac{\partial^2 B_{ccd}}{\partial d^2} &= -\kappa_{ccd} \left[2 \ln \left(\frac{d}{\hat{d}} \right) + \frac{2(d - \hat{d})}{d} + \frac{(d - \hat{d})(d + \hat{d})}{d^2} \right]. \end{aligned} \quad (45)$$

Similarly, the gradient and Hessian of B_{dcd} w.r.t. d are:

$$\frac{\partial B_{dcd}}{\partial d} = \begin{cases} 2\kappa_{dcd}(d - \hat{d}), & 0 \leq d < \hat{d}, \\ \kappa_{dcd}(-e^{-d} + 1 + d - 2\hat{d}), & d < 0, \end{cases} \quad (46)$$

and

$$\frac{\partial^2 B_{dcd}}{\partial d^2} = \begin{cases} 2\kappa_{dcd}, & 0 \leq d < \hat{d}, \\ \kappa_{dcd}(e^{-d} + 1), & d < 0. \end{cases} \quad (47)$$

Derivatives of medial distance are established via the medial state vector $\mathbf{p}$:

$$\frac{\partial d}{\partial \mathbf{q}} = \frac{\partial \mathbf{p}}{\partial \mathbf{q}} \frac{\partial d}{\partial \mathbf{p}}, \quad \frac{\partial^2 d}{\partial \mathbf{q}^2} = \left(\frac{\partial \mathbf{p}}{\partial \mathbf{q}} \right) \frac{\partial^2 d}{\partial \mathbf{p}^2} \left(\frac{\partial \mathbf{p}}{\partial \mathbf{q}} \right)^\top + \frac{\partial^2 \mathbf{p}}{\partial \mathbf{q}^2} \frac{\partial d}{\partial \mathbf{p}}. \quad (48)$$

The first- and second-order derivatives of d w.r.t. the medial vector $\mathbf{p}$ are computed by differentiating all the coefficients in Eq. (34) towards sphere centers and radii as in [Lan et al. 2021]. The second term $\frac{\partial^2 \mathbf{p}}{\partial \mathbf{q}^2} \frac{\partial d}{\partial \mathbf{p}} = \left(\frac{\partial^2 \tilde{\mathbf{q}}}{\partial \mathbf{q}^2} \frac{\partial \mathbf{p}}{\partial \tilde{\mathbf{q}}} + \frac{\partial \tilde{\mathbf{q}}}{\partial \mathbf{q}} \frac{\partial^2 \mathbf{p}}{\partial \tilde{\mathbf{q}}^2} \left(\frac{\partial \tilde{\mathbf{q}}}{\partial \mathbf{q}} \right)^\top \right) \frac{\partial d}{\partial \mathbf{p}}$ contains second-order derivatives of both the neural network and the coordinate transformation in Eq. (15). These contributions are generally indefinite and can adversely affect the robustness of the Newton solve. In our implementation, we therefore omit these higher-order terms and retain only the positive semi-definite Gauss-Newton approximation $\left(\frac{\partial \mathbf{p}}{\partial \mathbf{q}} \right) \frac{\partial^2 d}{\partial \mathbf{p}^2} \left(\frac{\partial \mathbf{p}}{\partial \mathbf{q}} \right)^\top$.

If we only have homogeneous affine deformation, $\mathbf{p}$ is linearly related to $\mathbf{q}_A$ as shown in Eq. (37) with a vanished Hessian. For vibrational deformation, neural MAT implicitly encodes the relation of $\mathbf{p}(\tilde{\mathbf{q}}_V)$. Thanks to its compact architecture, we analytically compute $\frac{\partial \mathbf{p}}{\partial \tilde{\mathbf{q}}_V}$ after the network is trained. Lastly, we take the network derivatives and differentiate Eq. (15) to back-propagate the derivatives to the world-frame coordinates $\mathbf{q}$ used in Eq. (17):

$$\frac{\partial \mathbf{p}}{\partial \mathbf{q}} = \frac{\partial \tilde{\mathbf{q}}}{\partial \mathbf{q}} \frac{\partial \mathbf{p}}{\partial \tilde{\mathbf{q}}}, \quad (49)$$

$$\frac{\partial \tilde{\mathbf{q}}}{\partial \mathbf{q}} = \begin{bmatrix} I_{12} & -\mathbf{v}_1 \text{diag}_4(\mathbf{A}^{-\top}) & \dots & -\mathbf{v}_r \text{diag}_4(\mathbf{A}^{-\top}) \\ \mathbf{0} & \text{diag}_4(\mathbf{A}^{-\top}) & & \mathbf{0} \\ \vdots & & \ddots & \\ \mathbf{0} & \mathbf{0} & & \text{diag}_4(\mathbf{A}^{-\top}) \end{bmatrix}, \quad (50)$$

where

$$\begin{aligned} \mathbf{v}_k &= [\mathbf{v}_{k,1}, \dots, \mathbf{v}_{k,12}]^\top \\ \mathbf{v}_{k,i+3j} &= \text{diag}_4(\mathbf{e}_{ij}) \tilde{\mathbf{q}}_{V_k}, \quad \forall i = 1, 2, 3, j = 1, 2, 3, 4 \\ \mathbf{e}_{ij} &= \begin{cases} \mathbf{e}_i \mathbf{e}_j^\top & \text{if } j < 4 \\ \mathbf{0}_{3 \times 3} & \text{if } j = 4 \end{cases}. \end{aligned}$$

B NETWORK DERIVATIVES

To enable stable time integration of collision constraints in Eq. (48), we compute the gradient of the medial parameters w.r.t. the generalized coordinates, namely $\frac{\partial \mathbf{p}}{\partial \tilde{\mathbf{q}}}$. To this end, we implemented a custom derivative kernel to efficiently compute the gradient. For each layer with a *sigmoid* activation function $\sigma(\cdot)$, we have the following relationship between output $\mathbf{y}$ and input $\mathbf{x}$ and their derivatives:

$$\mathbf{y} = \sigma(\mathbf{A}\mathbf{x} + \mathbf{b}), \quad \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = (\mathbf{S}\mathbf{A})^\top, \quad \mathbf{S} = \mathcal{D}(\sigma'(\mathbf{A}\mathbf{x} + \mathbf{b})), \quad (51)$$

where $\sigma'(x) = \sigma(x)(1 - \sigma(x))$ is the derivative of the *sigmoid* function. Let the FC layer weights be $\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3$ and activation layer derivatives be $\mathbf{S}_0, \mathbf{S}_1, \mathbf{S}_2$ respectively, we recursively apply the chain rule to obtain the derivative:

$$\frac{\partial \mathbf{p}}{\partial \tilde{\mathbf{q}}} = (\mathbf{A}_3 \mathbf{S}_2 \mathbf{A}_2 \mathbf{S}_1 \mathbf{A}_1 \mathbf{S}_0 \mathbf{A}_0)^\top. \quad (52)$$