Hyper-Dimensional Deformation Simulation

ALVIN SHI, HAOMIAO WU, and THEODORE KIM, Yale University, USA

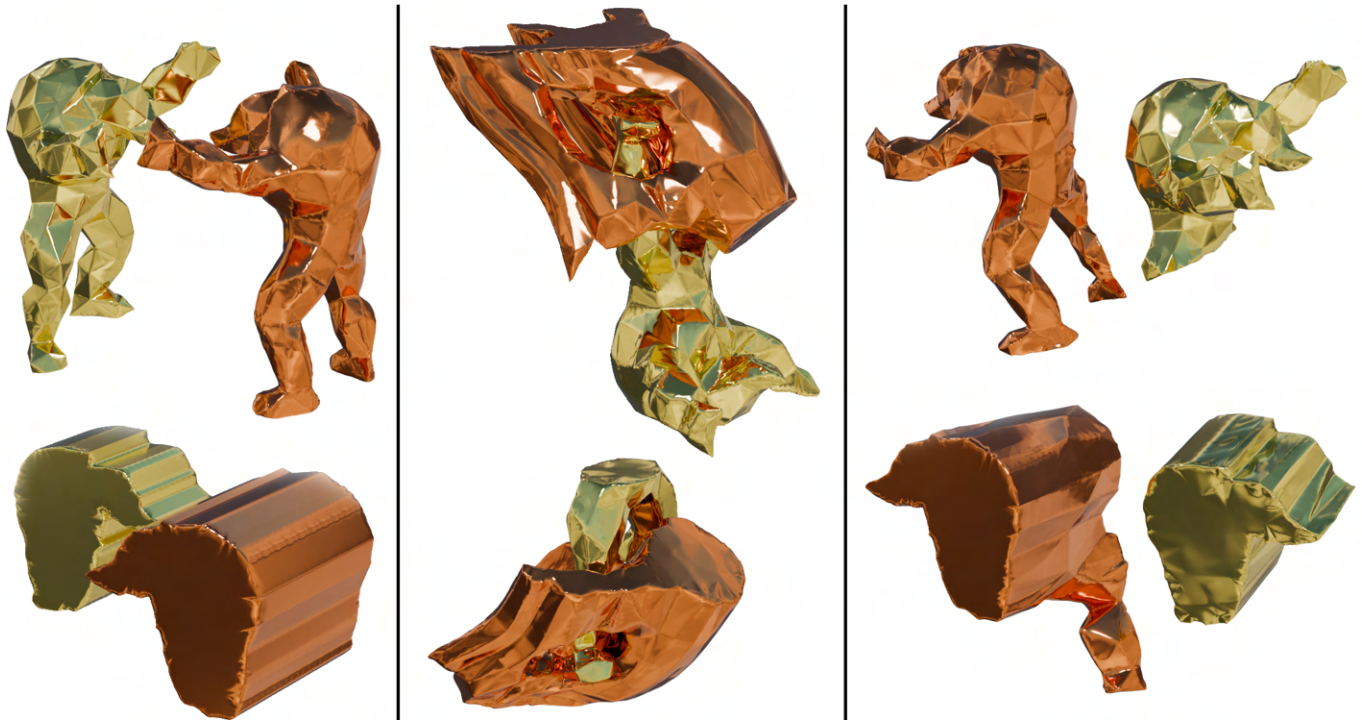

Fig. 1. Two 4D armadillo prisms brush past each other. In the top left, a 3D slice where the armadillos are most visible. Bottom left, a different 3D slice reveals the extrusion direction. In the middle are two stills from a four-dimensional rotation about their collision at frame 800. On the right, they have failed to hug and instead have passed through each other.

We present a method for simulating deformable bodies in four spatial dimensions. To accomplish this, we generalize several pieces of the traditional simulation pipeline. Starting from the meshing stage, we propose a simple method for generating a *pentachoral* mesh, the 4D analog of a tetrahedral mesh. Next, we show how to generalize the deformation invariants, allowing us to construct 4D hyperelastic energies that lead directly to hyper-dimensional deformation forces. Finally, we formulate collision detection and response in 4D. Our eigenanalyses of the resulting deformation and collision energies generalize to arbitrarily higher dimensions. The resulting simulations display a variety of previously unseen visual phenomena.

CCS Concepts: • **Computing methodologies** → *Collision detection*; **Physical simulation**; *Mesh geometry models*; **Model development and analysis**.

Additional Key Words and Phrases: Physical Simulation, N-Dimensional Physics, Collision Detection

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGGRAPH Conference Papers '25, August 10–14, 2025, Vancouver, BC, Canada
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1540-2/2025/08.
https://doi.org/10.1145/3721238.3730730

ACM Reference Format:
Alvin Shi, Haomiao Wu, and Theodore Kim. 2025. Hyper-Dimensional Deformation Simulation. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25), August 10–14, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3721238.3730730

1 INTRODUCTION

Hyper-dimensional geometries, 4D shapes that occupy four spatial dimensions beyond the traditional 2D and 3D, have attracted interest in art [Henderson 2018], visualization [Hanson and Cross 1993], perception [Miwa et al. 2015], film [Seymour 2014] and VR [Jamroz 2009]. Cavallo [2021] surveyed previous computer graphics work, which includes the rendering of higher-dimensional objects [Kim et al. 2022] and their rigid body simulation [Bosch 2020].

We present the first hyper-dimensional simulation of 4D *deformation*. This requires generalizing many pieces of the simulation pipeline, including meshing, force computation, and collision processing. We propose novel approaches to all of these, and demonstrate their effectiveness in a variety of scenes.

For meshing, the 4D analog of a 3D tetrahedral mesh is a *pentachoral* mesh. While methods for generating 3D meshes are mature,

4D meshing techniques are far less developed. We present a simple extrusion and filling method that uses existing 3D meshing techniques to generate viable 4D meshes.

For internal material forces, we show how to generalize the traditional 2D and 3D deformation invariants [Bonet et al. 2021]. Once these are in place, existing hyperelastic energies can be applied in 4D. The eigenanalysis of these invariants [Lin et al. 2024; Smith et al. 2019] can then be extended to 4D, which additionally reveals the eigenstructure of the arbitrary n-D case.

For collision processing, we show that the equivalent of point-triangle and edge-edge collisions in 3D are *point-tetrahedron* and *edge-triangle* in 4D. We formulate collision penalty energies for both of these cases, and similar to the 3D case [Huang et al. 2024; Shi and Kim 2023], are able to obtain analytic eigensystems for each. Our analysis again generalizes to n-D, effectively obtaining the eigensystems for *all* higher-dimensional penalty energies.

Our contributions are:

- The first hyper-dimensional deformable body simulator.
- An extrusion and filling method for 4D mesh generation.
- 4D formulation of elastic deformation and collision energies.
- n-D analysis and eigensystems for elastic deformation and collision energies.

2 RELATED WORK

Deformation in computer graphics has traditionally been examined in 2D [Alexa et al. 2000] and 3D [Baraff and Witkin 1998]. One popular method [Debunne et al. 2001; Müller et al. 2002] is to simulate a 3D volume using a tetrahedral mesh [Alliez et al. 2005; Hu et al. 2018; Si 2015], resolve its internal material forces using hyperelastic strain energies [Bonet et al. 2021], and to resolve collisions using a variety of energy-based and geometric methods [Andrews et al. 2022; Gottschalk et al. 1996; Moore and Wilhelms 1988].

This class of physics-based simulations has not previously been extended to 4D, although other topics in computer graphics have been investigated in higher dimensions. One of the most familiar is the representation of 3D rotations using (unit) 4D quaternions [Hanson 2005]. Quaternions have also been used to generate hyper-dimensional fractals [Hart et al. 1989; Norton 1982], but did not simulate elastic deformation. Other 4D approaches assign novel sensory modalities to the fourth dimension [Nam et al. 2024], but we explicitly assign it to a spatial dimension.

Existing work has instead largely focused on rendering and visualization. For example, van Wijk and van Liere [1993] proposed the *Hyperslice* method for extracting 2D slices from higher-dimensional data, while Hibbard et al. [1996] introduced the *Vis5D* system for visualizing time-varying 4D scalar data. More recently, Kim et al. [2022] proposed a method for modeling and rendering non-Euclidean geometries, in particular 3D polytopes embedded in 4D Euclidean spaces, and Cavallo [2021] examined the authoring the rendering issues associated with 4D spatial content.

3D meshing can use higher dimensions to remove self-intersections [Zhong et al. 2018] and ensure anisotropy [Lévy and Bonneel 2013], but the problem of full 4D meshing has only been investigated by a handful of researchers [Brandts et al. 2007; Petrov and Todorov 2021], and sometimes attaches the additional dimension to time [Caplan et al. 2020] instead of space.

Other fields have investigated high-dimensional elasticity for several applications. Borrel and Bechmann [1991] use the fourth dimension as a space-time regularizer, and Zhao [2017] use four-dimensional elastic lattices to surpass limitations in three-dimensional models. von Danwitz et al. [2021] additionally represent topology changes for obstacles in fluid dynamics with four-dimensional elastic meshes, and Wang et al. [2016] formalizes relativistic deformation using spacetime coordinates.

The closest work to ours is Bosch [2020], which formulated rigid body dynamics in higher dimensions. They also reformulated the collision pipeline to account for the extra dimension, but the rigidity condition allowed them to address overlapping polytopes. More granular point-tetrahedron and edge-triangle cases are required under deformation. Their rigidity assumption also only required a 4D inertia tensor to enable dynamics simulation, but deformation requires a full 4D elasticity formulation.

3 MESH GENERATION

In 2D, the simplicial element is the triangle, while in 3D the equivalent simplex is the tetrahedron. In 4D, the equivalent primitive is the 5-point *pentachoron*. Unlike triangle and tetrahedral meshes, pentachoral meshes are not readily available online, and there is no standard meshing algorithm for their generation.

We will instead use well-conditioned tetrahedral meshes generated using mature algorithms [Si 2015] as a starting point to create simulation-ready pentachoralized prismatic meshes. We present the process as a sequence of extrusions and fillings. For our use case, these techniques provide a sufficient amount of control over mesh resolution and expressiveness. Higher-dimensional constrained Delaunay triangulations are also possible, but adding Steiner points for these cases is still challenging [Shewchuk 2008].

3.1 3D Extrude and Fill

For illustrative purposes, we will first show the extrusion and filling process in 3D, where we extrude a triangle mesh to generate a tetrahedral mesh. With this established, we can then extend the process analogously to 4D, where we extrude a tetrahedral mesh into a pentachoral mesh.

Consider a set of mesh vertices $\mathcal{V} = \{\mathbf{v}_0, \ldots, \mathbf{v}_n\}$, where $\mathbf{v}_i \in \mathbb{R}^3$. Let a triangle $\mathbf{t} = \{i, j, k\}$ be a triplet of integers that specify $\mathbf{v}_i$, $\mathbf{v}_j$, and $\mathbf{v}_k$ as a triangle in the mesh. As input, we take a set of vertices $\mathcal{V}$ and triangles $\mathcal{T} = \{\mathbf{t}_0, \ldots, \mathbf{t}_m\}$, along with an extrusion vector $\mathbf{v}_\mathrm{e} \in \mathbb{R}^3$. We can then generate a tetrahedral mesh along the extrusion direction by forming a new set of vertices $\mathcal{V}_\mathrm{e}$ and tetrahedra $\mathcal{H}_\mathrm{e} \subset \mathbb{N}^4$.

3.1.1 Single triangle preliminaries. The case of a single triangle is straightforward. Given $\mathcal{V} = \{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2\}$, $\mathcal{T} = \{\{0, 1, 2\}\}$, and extrusion direction $\mathbf{v}_\mathrm{e}$, a set of extruded vertices can be written:

$$\mathcal{V}_\mathrm{e} = \{\mathbf{v}_0,\ \mathbf{v}_1,\ \mathbf{v}_2,\ \mathbf{v}_0 + \mathbf{v}_\mathrm{e},\ \mathbf{v}_1 + \mathbf{v}_\mathrm{e},\ \mathbf{v}_2 + \mathbf{v}_\mathrm{e}\} \tag{1}$$

The new set $\mathcal{V}_\mathrm{e}$ now forms a prism that must then be decomposed into tetrahedra. Successive corner cutting leads to one possible

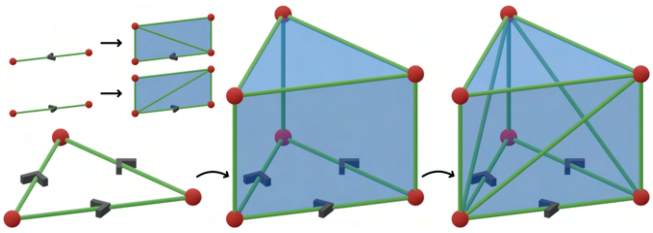

Fig. 2. *Correspondence from directed graphs to surface triangulation.* For each edge on the input shape, we cut a diagonal on the corresponding extruded face according to the direction assigned to the edge.

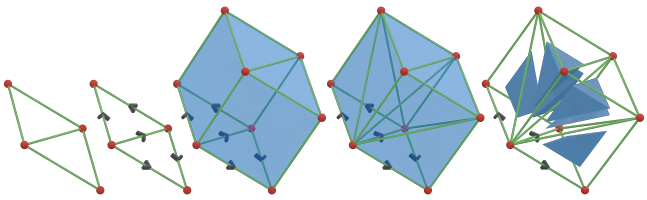

Fig. 3. *Extrusion/tetrahedralization with a simple starting mesh.* First create a directed acyclic graph (DAG) using vertex orders, then assign corresponding tetrahedralizations to each triangular prism. The same process generates pentachoralized models from tetrahedral meshes.

decomposition:

$$\mathcal{H}_e = \{\{0,1,2,5\},\{0,3,4,5\},\{0,1,4,5\}\} \tag{2}$$

Rotations and reflections of vertex labels then yield six distinct possible decompositions. For the single triangle case, selecting any one of the decompositions suffices to generate a tetrahedralized triangular prism.

3.1.2 Multiple triangles. For adjacent triangles that share edges, obtaining $\mathcal{V}_\mathbf{e}$ remains straightforward: add $\mathbf{v}_\mathbf{e}$ to each $\mathbf{v}_i \in \mathcal{V}$ and append to $\mathcal{V}_\mathbf{e}$. However, tetrahedral decomposition becomes more involved. The shared edges between triangles extrude into rectangles that must be triangulated in a way that ensures compatibility between each prism pair.

To guarantee compatibility, we observe that each tetrahedralization of the triangular prism induces a unique set of boundary cuts on the prism's surface, where the diagonal cut on each rectangle can then be encoded as a direction on the edge of the original triangle (Fig. 2). By assigning each edge a direction, we guarantee that boundaries between tetrahedralized prisms are compatible after extrusion. Adjacent prisms query their shared edge's direction to determine the orientation of the diagonal cut, so there is no possibility of having mismatching cuts from incompatible decompositions.

However, it can be impossible to cleanly tetrahedralize prisms with certain sets of boundary cuts without Steiner points. Fortunately, the only two cases where an extruded triangle tetrahedralization requires Steiner points is exactly the case of a cycle. As long as edge directions form a directed acyclic graph (DAG), the diagonals encoded by the directions form a consistent, compatible tetrahedralization of the adjoining prisms.

Using vertex indices as a total order to obtain a DAG, we then assign every extruded prism a tetrahedralization according to the edge directions, which yields a tetrahedral mesh with consistent internal faces (Fig. 3). We do not claim this as a general method for generating tetrahedral meshes, but rather *one* way of reliably generating a viable mesh. Crucially, the method generalizes to 4D.

3.2 4D Extrude and Fill

Extrusion in 4D follows from the 3D case. We start with a tetrahedral mesh composed of vertices $\mathcal{V}$, where $\mathbf{v}_i \in \mathbb{R}^4$ with the fourth coordinate set to zero, tetrahedra $\mathcal{H} \subset \mathbb{N}^4$, and an extrusion direction $\mathbf{v}_\mathbf{e} \in \mathbb{R}^4$ where the fourth coordinate is non-zero. We can extrude $\mathcal{V}$ along $\mathbf{v}_\mathbf{e}$ to obtain $\mathcal{V}_\mathbf{e} \subset \mathbb{R}^4$, which forms a set of 4D tetrahedral prisms that must then be decomposed into pentachorons.

Analogous to the 3D case, where each rectangular face must be divided in a way that guarantees that the resulting tetrahedra are compatible, each tetrahedral prism must now be divided in a way to ensure compatible pentachora. Fortunately, the strategy of forming a DAG and using the edge directions to prescribe boundary decompositions continues to work in 4D.

By examining the edge directions of a tetrahedral prism, we can assign it one of 24 possible pentachoral decompositions. The DAG on the initial tetrahedral mesh will then guarantee cross-prism compatibility. In the supplemental materials, we have included scripts that exhaustively ensure that every possible set of boundary cuts assigned to a tetrahedron in this manner gives rise to a valid pentachoralization. We also include programs for performing extrusion/filling in 3D and 4D.

Others have speculated that finding a valid pentachoralization of a 3D mesh extruded into a 4D prism is equivalent to solving the NP-complete monochromatic triangle problem on a graph [CodeParade 2023]. We have found that it instead reduces to the much simpler problem of constructing a DAG.

Finally, even if the initial tetrahedral mesh was well-conditioned [Shewchuk 2002], there is no guarantee that the resulting pentachoral mesh will be as well. To the contrary, if $\|\mathbf{v}_e\|$ is very large, it will inevitably generate "skinny" pentachorons that are degenerate along one direction, and lead to catastrophically ill-conditioned matrix inverses during deformation gradient computation. [Kim et al. 2019]

To ensure that the pentachorons were well-conditioned, if $\|\mathbf{v}_e\|$ was large, we subdivided the vector and performed the extrusion in several smaller steps rather than one large displacement. An analogous 3D surface operation would be be subdivide a cylinder along its height in order to prevent long, skinny triangles from stretching from one end cap to another. Generalized subdivision is also available for refinement purposes [Petrov and Todorov 2018].

4 DEFORMATION SIMULATION

To simulate deformations in 4D, we can use any standard method, e.g. the implicit solver from Baraff and Witkin [1998]:

$$\left(\mathbf{M} - h\frac{\partial \mathbf{f}}{\partial \mathbf{v}} - h^2\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)\Delta\mathbf{x} = h^2\mathbf{f} + h\left(\mathbf{M} - h\frac{\partial \mathbf{f}}{\partial \mathbf{v}}\right)\mathbf{v}_n \tag{3}$$

Above, $\mathbf{M}$ is the mass matrix, $\mathbf{f}$ is the internal material force, h is the timestep, and $\mathbf{x}$ and $\mathbf{v}$ are the position and velocity. The overall form of the equation is agnostic to the spatial dimension, and remains the same between 2D and 3D. Thus, it can also be applied in 4D, but requires a 4D method for computing $\mathbf{f}$ and its derivatives.

For concreteness, we will examine a 4D version of $\mathbf{f}$ using the Stable Neo-Hookean (SNH) [Smith et al. 2018] elastic energy

$$\psi_{\text{SNH}} = \frac{1}{2}\mu(I_2 - 4) + \frac{1}{2}\lambda(I_3 - \alpha)^2 \tag{4}$$

where μ is the shear modulus, or Lame's second parameter, λ is Lame's first parameter, and $\alpha = 1 + \frac{\mu}{\lambda}$. The terms

$$I_2 = \text{tr}(\mathbf{F}^\top\mathbf{F}) \qquad I_3 = \det(\mathbf{F}) \tag{5}$$

are deformation invariants [Bonet et al. 2021], which are also agnostic to the spatial dimension. The $\mathbf{F}$ denotes the deformation gradient.

Sections 4.1 through 4.5 contain our detailed analyses of higher-dimensional invariants and their derivatives in 4D, allowing for energy Hessian eigenclamping. For readers interested in collision detection, we refer them to section 5.

4.1 Deformation Gradient

The size of the deformation gradient matrix corresponds to the spatial dimension. In 2D, $\mathbf{F} \in \mathbb{R}^{2\times2}$. In 3D, $\mathbf{F} \in \mathbb{R}^{3\times3}$. It follows that in 4D, $\mathbf{F} \in \mathbb{R}^{4\times4}$. Computing $\mathbf{F}$ also follows analogously from the 2D and 3D cases. Given a pentachoron composed of the rest points $\{\bar{\mathbf{p}}_0, ..., \bar{\mathbf{p}}_4\}$ and deformed points $\{\mathbf{p}_0, ..., \mathbf{p}_4\}$, we can construct:

$$\mathbf{D}_m = \left[\begin{array}{c|c|c|c} \bar{\mathbf{p}}_1 - \bar{\mathbf{p}}_0 & \bar{\mathbf{p}}_2 - \bar{\mathbf{p}}_0 & \bar{\mathbf{p}}_3 - \bar{\mathbf{p}}_0 & \bar{\mathbf{p}}_4 - \bar{\mathbf{p}}_0 \end{array}\right] \tag{6}$$

$$\mathbf{D}_s = \left[\begin{array}{c|c|c|c} \mathbf{p}_1 - \mathbf{p}_0 & \mathbf{p}_2 - \mathbf{p}_0 & \mathbf{p}_3 - \mathbf{p}_0 & \mathbf{p}_4 - \mathbf{p}_0 \end{array}\right] \tag{7}$$

The expression for $\mathbf{F}$ and its labeled columns is then:

$$\mathbf{F} = \mathbf{D}_s\mathbf{D}_m^{-1} = \left[\begin{array}{c|c|c|c} \mathbf{f}_0 & \mathbf{f}_1 & \mathbf{f}_2 & \mathbf{f}_3 \end{array}\right] \tag{8}$$

With $\mathbf{F}$ established, we can now analyze the deformation invariants.

4.2 I_2 Eigenanalysis

For the $I_2 = \text{tr}(\mathbf{F}^\top\mathbf{F})$ invariant, the general form of the gradient is the same as in 2D and 3D, albeit in $\mathbb{R}^{16}$:

$$\frac{\partial I_2}{\partial \mathbf{F}} = 2\mathbf{F} \qquad \mathbf{g}_2 = \text{vec}\left(\frac{\partial I_2}{\partial \mathbf{F}}\right) \tag{9}$$

and the vectorized $\mathbb{R}^{16\times16}$ force gradient is

$$\text{vec}\left(\frac{\partial^2 I_2}{\partial \mathbf{F}^2}\right) = \frac{\partial \mathbf{g}_2}{\partial \mathbf{F}} = \mathbf{H}_2 = 2\mathbf{I} \tag{10}$$

Similar to the 2D and 3D cases, the eigenvalues are all 2, and the multiplicity means that the eigenvectors span an arbitrary rank-16 vector space. The form of $\mathbf{H}_2$ will stay consistent regardless of the dimension, so the n-D eigensystem is always n^2 eigenvalues equal to 2, and an arbitrary rank-n^2 vector space.

4.3 I_3 Eigenanalysis

4.3.1 4D Cross Product. The 3D cross product of two vectors generates a third orthogonal vector that completes the basis span. An analogous 4D operation should take three vectors and produce a fourth orthogonal vector. We can generalize the determinant-like formulation of the 3D cross product as

$$\times(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \det\begin{bmatrix} x_0 & y_0 & z_0 & \hat{\mathbf{e}}_0 \\ x_1 & y_1 & z_1 & \hat{\mathbf{e}}_1 \\ x_2 & y_2 & z_2 & \hat{\mathbf{e}}_2 \\ x_3 & y_3 & z_3 & \hat{\mathbf{e}}_3 \end{bmatrix} \tag{11}$$

where $\mathbf{e}_i$ are the principle directors of $\mathbb{R}^4$. Carrying through expansion by minors and using scalar-vector multiplication yields the following vector in $\mathbb{R}^4$:

$$\begin{aligned}\times(\mathbf{x}, \mathbf{y}, \mathbf{z}) = &-\hat{\mathbf{e}}_0 \det\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} + \hat{\mathbf{e}}_1 \det\begin{bmatrix} x_0 & y_0 & z_0 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \\ &-\hat{\mathbf{e}}_2 \det\begin{bmatrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ x_3 & y_3 & z_3 \end{bmatrix} + \hat{\mathbf{e}}_3 \det\begin{bmatrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{bmatrix}\end{aligned}$$

For any linearly independent triplet $\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$, $\times(\mathbf{x}, \mathbf{y}, \mathbf{z})$ generates a fourth orthogonal vector. We additionally define the linear operator $[\mathbf{u}, \mathbf{v}]_\times \in \mathbb{R}^{4\times4}$ such that $[\mathbf{u}, \mathbf{v}]_\times \cdot \mathbf{x} = \times(\mathbf{u}, \mathbf{v}, \mathbf{x})$:

$$[\mathbf{u}, \mathbf{v}]_\times = \begin{bmatrix} 0 & u_3v_2 - u_2v_3 & u_1v_3 - u_3v_1 & u_2v_1 - u_1v_2 \\ u_2v_3 - u_3v_2 & 0 & u_3v_0 - u_0v_3 & u_0v_2 - u_2v_0 \\ u_3v_1 - u_1v_3 & u_0v_3 - u_3v_0 & 0 & u_1v_0 - u_0v_1 \\ u_1v_2 - u_2v_1 & u_2v_0 - u_0v_2 & u_0v_1 - u_1v_0 & 0 \end{bmatrix}$$

4.3.2 I_3 Derivatives. Using these operators, the gradient becomes

$$\frac{\partial I_3}{\partial \mathbf{F}} = \left[\begin{array}{c|c|c|c} -\times(\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3) & \times(\mathbf{f}_0, \mathbf{f}_2, \mathbf{f}_3) & -\times(\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_3) & \times(\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2) \end{array}\right]$$

Using the vectorization $\mathbf{g}_3 = \text{vec}\left(\frac{\partial I_3}{\partial \mathbf{F}}\right)$, the Hessian becomes

$$\frac{\partial \mathbf{g}_3}{\partial \mathbf{F}} = \mathbf{H}_3 = \begin{bmatrix} 0 & [\mathbf{f}_3, \mathbf{f}_2]_\times & [\mathbf{f}_1, \mathbf{f}_3]_\times & [\mathbf{f}_2, \mathbf{f}_1]_\times \\ \ddots & 0 & [\mathbf{f}_3, \mathbf{f}_0]_\times & [\mathbf{f}_0, \mathbf{f}_2]_\times \\ \ddots & \ddots & 0 & [\mathbf{f}_1, \mathbf{f}_0]_\times \\ \text{sym.} & \dots & \dots & 0 \end{bmatrix}$$

We can now analyze the invariant's eigensystem.

4.3.3 Eigensystem. The 16 eigenpairs of $\mathbf{H}_3$ are 4D generalizations of the 3D eigenpairs for I_3 from Smith et al. [2018]. Given the singular value decomposition $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}^\top$, the first six eigenvalues correspond to generalized *twist* modes. For brevity, we list the first two here, with the complete listing in §1 of the supplemental materials:

$$\lambda_0^{I_3} = \sigma_0\sigma_1 \qquad \lambda_1^{I_3} = \sigma_0\sigma_2 \tag{12}$$

The σ_i are singular values from Σ. The eigenmatrices $\mathbf{Q}_i$ vectorize into the eigenvectors $\mathbf{q}_i = \text{vec}(\mathbf{Q}_i)$, and have the general form

$\mathbf{Q}_i = {}^1/\sqrt{2}\mathbf{U}\Theta_i\mathbf{V}^\top$. The first two Θ_i are

$$\Theta_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \qquad \Theta_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \tag{13}$$

and the complete set is in the supplemental materials. The next six are generalized *flip* modes, where $\lambda^{I_3}_{i+6} = -\lambda^{I_3}_i$. The first two eigenmatrices are

$$\Theta_6 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \Theta_7 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{14}$$

The final four eigenmodes are linear combinations of the *stretch* modes, the first two of which are:

$$\mathbf{Q}'_0 = \mathbf{U}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\mathbf{V}^\top \qquad \mathbf{Q}'_1 = \mathbf{U}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\mathbf{V}^\top \tag{15}$$

Since they are orthogonal to the other twelve eigenmodes, we can deflate $\mathbf{H}_3$ using $\mathbf{q}'_i = \mathrm{vec}\left(\mathbf{Q}'_i\right)$ and $\mathbf{D} = \left[\mathbf{q}'_0|\mathbf{q}'_1|\mathbf{q}'_2|\mathbf{q}'_3\right]$ to obtain

$$\mathbf{D}^\top\mathbf{H}_3\mathbf{D} = \begin{pmatrix} 0 & \sigma_2\sigma_3 & \sigma_1\sigma_3 & \sigma_1\sigma_2 \\ \sigma_2\sigma_3 & 0 & \sigma_0\sigma_3 & \sigma_0\sigma_2 \\ \sigma_1\sigma_3 & \sigma_0\sigma_3 & 0 & \sigma_0\sigma_1 \\ \sigma_1\sigma_2 & \sigma_0\sigma_2 & \sigma_0\sigma_1 & 0 \end{pmatrix} \tag{16}$$

Similar to Smith et al. [2018], the eigenpairs of this deflated matrix can be used to obtain the final four modes of $\mathbf{H}_3$.

4.3.4 n-D Generalization. The analytical derivatives of I_3 can be stated for $n \geq 2$ using Jacobi's formula:

$$\frac{\partial I_3}{\partial \mathbf{F}} = \frac{\partial \det(\mathbf{F})}{\partial \mathbf{F}} = (\mathrm{adj}\,(\mathbf{F}))^\top \qquad (\mathbf{H}_3)_{ijpq} = (-1)^s\mathbf{M}_{ip,jq}$$

where $0 \leq i, j, p, q \leq n-1$ and adj $(\mathbf{F})$ is the adjugate matrix of $\mathbf{F}$. The matrix $\mathbf{M}_{ip,jq}$ is the second minor: the determinant of $\mathbf{F}$ with its ith and pth row, the jth and qth column removed, and $\mathbf{M}_{ip,jq} = 0$ when $i = p$ or $j = q$. The constant s is determined by:

$$s = \begin{cases} i+j+p+q & (i < p \text{ and } j < q) \text{ or } (i > p \text{ and } j > q) \\ i+j+p+q+1 & \text{otherwise} \end{cases}$$

For the eigensystems, following the patterns spanning 2D to 4D, we hypothesize that any n-D version of I_3 will yield n linearly combined stretch modes. The remaining $n^2 - n$ modes will be half twist modes and half flip modes. Since $I_3 = \Pi_{i=0}^{n-1}\sigma_i$, the corresponding flip eigenvalues will be $-I_3/\sigma_i\sigma_j$, where $i \neq j$ and 1s appear at (i, j) and (j, i) in the center matrix. In the twist eigenpairs, a 1 gets negated.

For the stretch modes, we observe that the 2D and 3D cases show a pattern of deflations under $\mathbf{D}^\top\mathbf{H}_3\mathbf{D}$:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \text{ (in 2D)} \Rightarrow \begin{pmatrix} 0 & \sigma_2 & \sigma_1 \\ \sigma_2 & 0 & \sigma_0 \\ \sigma_1 & \sigma_0 & 0 \end{pmatrix} \text{ (in 3D)} \tag{17}$$

Combined with Eqn. 16, we hypothesize that for arbitrary n-D stretch modes, the generalized deflated scaling matrix is $\partial^2 I_3/\partial(\mathrm{vec}(\Sigma))^2$, which can be expressed as:

$$\left(\mathbf{D}^\top\mathbf{H}_3\mathbf{D}\right)_{ij} = \begin{cases} I_3/\sigma_i\sigma_j & i \neq j \\ 0 & i = j \end{cases} \tag{18}$$

The scaling modes can then be isolated via eigendecomposition. We have numerically verified that our hypothesis holds for up to 100 spatial dimensions. Code is provided in our supplemental materials.

4.4 I_1 Eigenanalysis

The I_1 invariant does not appear in the SNH energy, but we present its n-D eigenanalysis here so that any 3D isotropic energy, e.g. the ARAP energy [Sorkine and Alexa 2007], can be rewritten in n-D.

If $\mathbf{F} = \mathbf{RS}$ is the polar decomposition, then we can denote $I_1 = \mathrm{tr}(\mathbf{S})$. The gradient of I_1 generalizes from 3D to higher dimensions using $\mathrm{tr}(\mathbf{S}) = \mathrm{tr}(\mathbf{F}^\top\mathbf{R})$, which yields $\partial I_1/\partial\mathbf{F} = \mathbf{R}$. The Hessian is then the n-D *rotation gradient*, $\mathbf{H}_1 = \partial\mathbf{R}/\partial\mathbf{F}$. Similar to previous works [Smith et al. 2019], we assemble it from its eigenpairs.

The n-D form of $\mathbf{H}_1$ has $(n^2-n)/2$ non-zero eigenpairs. For any $i, j \in \{0, 1, \cdots, n-1\}$ and $i < j$, there exists an eigenpair:

$$\lambda_{ij} = \frac{2}{\sigma_i + \sigma_j} \qquad \mathbf{Q}_{ij} = \frac{1}{\sqrt{2}}\mathbf{U}\Theta\mathbf{V}^\top \tag{19}$$

where $\Theta \in \mathbb{R}^{n\times n}$, with entries $\Theta(i, j) = 1$, $\Theta(j, i) = -1$ and all other entries are zero. The generalized rotation gradient is then:

$$\mathbf{H}_1 = \sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1}\lambda_{ij}\mathrm{vec}\left(\mathbf{Q}_{ij}\right)\left(\mathrm{vec}\left(\mathbf{Q}_{ij}\right)\right)^\top \tag{20}$$

We have numerically verified that this expression holds for up to 100 spatial dimensions, and code is provided in the supplement.

4.5 4D Stable Neo-Hookean

With the invariant analysis complete, we can now compute the eigensystem of the SNH energy. The Hessian matches the general form from 2D and 3D:

$$\frac{\partial^2\psi_{\mathrm{SNH}}}{\partial\mathbf{F}^2} = \mu\mathbf{I} + \lambda\left((I_3 - \alpha)\frac{\partial^2 I_3}{\partial\mathbf{F}^2} + \mathbf{g}_3\mathbf{g}_3^\top\right) \tag{21}$$

The first dozen eigenvectors are exactly the twist and flip modes, and the eigenvalues are

$$\lambda_i^{\mathrm{SNH}} = \lambda(I_3 - \alpha)\lambda_i^{I_3} + \mu \tag{22}$$

which matches the general 3D form from Kim and Eberle [2022], but with the extra dimension adding a new σ_* to each $\lambda_i^{I_3}$. For the final four eigenmodes, we build the remaining subspace spanned by the scaling modes by treating it as a rank-one update to the reduced matrix in Eqn. 16. We compute the four modes with a numerical solver.

Equipped with eigensystems for the invariants I_1, I_2, and I_3 in any dimension, general hyper-dimensional deformation energy eigenanalysis follows. For more details, see §2 in the supplement.

5 COLLISION PROCESSING

As dimensionality increases, so does the the number of collision primitive pairs. 2D has point-edge collisions, while 3D has point-face and edge-edge collisions. For a dimension n, the number of primitive pairs corresponds to the number of ways that $n+1$ points can be split into two simplices: $\lceil n/2 \rceil$. Thus, in 4D we again have two cases to consider: point-tetrahedron and edge-triangle pairs.

5.1 Point-Tetrahedron Collisions

For point-tetrahedron pairs $\mathbf{x}_c$, $\{\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3\}$, we first find the tetrahedral barycentric coordinates $\{\alpha_0, \alpha_1, \alpha_2, \alpha_3\}$ of the closest interior point to the colliding vertex:

$$\underset{\substack{\alpha_{0\ldots3} \\ 0 \le \alpha_{0\ldots3} \le 1}}{\operatorname{argmin}} \left\| \mathbf{x}_c - \sum_{i=0}^{3} \alpha_i \mathbf{t}_i \right\| \tag{23}$$

Solving for the α_is, we then denote unsigned distance between a collision point $\mathbf{x}_c$ and the closest point on the tetrahedron $\sum_{i=0}^{3} \alpha_i \mathbf{t}_i$ as l. Afterwards, we insert a spring force whenever a distance threshold ϵ is crossed, giving rise to a penalty energy

$$\psi_{\text{pt}}(l) = \begin{cases} k_c(sl - \epsilon)^2 & l < \epsilon \\ 0 & l \ge \epsilon \end{cases} \tag{24}$$

where k_c is a spring constant and $s \in \{-1, 1\}$ is set based on whether $\mathbf{x}_c$ is outside or inside the surface tetrahedron's pentachoron.

5.2 Edge-Triangle Collisions

For edge-triangle pairs $\{\mathbf{e}_0, \mathbf{e}_1\}$, $\{\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2\}$, we derive collision weights from a modified distance-finding minimization:

$$\underset{\substack{\beta_i, \alpha_n \\ 0 \le \beta_i, \alpha_n \le 1}}{\operatorname{argmin}} \left\| \sum_{i=0}^{1} \beta_i \mathbf{e}_i - \sum_{n=0}^{2} \alpha_n \mathbf{t}_n \right\| \tag{25}$$

Taking the minimized value as the edge-triangle length l and using another spring force, we obtain the energy

$$\psi_{\text{et}}(l) = \begin{cases} k_c(l - \epsilon)^2 & l < \epsilon \\ 0 & l \ge \epsilon \end{cases} \tag{26}$$

where k_c and ϵ are the same as in Eqn. 24.

5.3 Generalized n-D Collision Analysis

In line with previous analyses [Huang et al. 2024; Shi and Kim 2023], we can derive a general eigensystem for n-D length-based penalties. We will begin with the simple case of a single vertex against a set of vertices, then expand to any two vertex sets.

5.3.1 Point-simplex length analysis. Consider a point $\mathbf{x}_c \in \mathbb{R}^n$ and a collection of k points in $\mathbb{R}^n$ denoted as $\{\mathbf{p}_0, ..., \mathbf{p}_{k-1}\}$ with convex hull $\mathcal{F}$. Let $\mathbf{x}_p = \sum_{i=0}^{k-1} \alpha_i \mathbf{p}_i$. For collisions, α_i are the barycentric coefficients that represent the point on $\mathcal{F}$ that is closest to $\mathbf{x}_c$, though no assumptions on α_i are needed in the following analysis.

We begin by taking derivatives of the general unsigned length $l_u = \|\mathbf{x}_c - \mathbf{x}_p\|$ by first defining intermediates:

$$\mathbf{v} = \begin{bmatrix} \mathbf{x}_c \\ \mathbf{p}_0 \\ \vdots \\ \mathbf{p}_{k-1} \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} 1 \\ -\alpha_0 \\ \vdots \\ -\alpha_{k-1} \end{bmatrix} \quad \mathbf{P} = \mathbf{I} - \frac{(\mathbf{x}_c - \mathbf{x}_p)(\mathbf{x}_c - \mathbf{x}_p)^\top}{\|\mathbf{x}_c - \mathbf{x}_p\|^2} \tag{27}$$

The derivatives are then

$$\frac{\partial l_u}{\partial \mathbf{v}} = \mathbf{g}_l = \frac{1}{l_u} \mathbf{w} \otimes (\mathbf{x}_c - \mathbf{x}_p) \tag{28}$$

$$\frac{\partial^2 l_u}{\partial \mathbf{v}^2} = \mathbf{H}_l = \frac{1}{l_u} \mathbf{w}\mathbf{w}^\top \otimes \mathbf{P} \tag{29}$$

where $\otimes$ is the Kronecker product. The Kronecker product implies that the eigensystem of $\mathbf{H}_l$ reduces to the eigensystems of $\mathbf{w}\mathbf{w}^\top$ and $\mathbf{P}$. Since $\mathbf{P}$ is a projection matrix into the subspace orthogonal to $\mathbf{x}_c - \mathbf{x}_p$ (denoted as $(\mathbf{x}_c - \mathbf{x}_p)^\perp$), its eigensystem is:

$$\lambda_j^{\mathbf{P}} = 1 \qquad \mathbf{q}_j^{\mathbf{P}} \in (\mathbf{x}_c - \mathbf{x}_p)^\perp \tag{30}$$

Since $\mathbf{w}\mathbf{w}^\top$ is an outer product, it has a single nontrivial eigenvector:

$$\lambda_0^{(\mathbf{w}\mathbf{w}^\top)} = 1 + \sum_{n=0}^{k-1} \alpha_n^2 \qquad \mathbf{q}_0^{(\mathbf{w}\mathbf{w}^\top)} = \mathbf{w} \tag{31}$$

The eigenpairs of $\mathbf{H}_l$ are then constructed by selecting any two eigenpairs from $\mathbf{w}\mathbf{w}^\top$ and $\mathbf{P}$, multiplying their eigenvalues, and taking the Kronecker product of their eigenvectors. The pairs with non-zero eigenvalue then take the form

$$\lambda^{\mathbf{H}_l} = \frac{1}{l_u} \left(1 + \sum_{n=0}^{k-1} \alpha_n^2 \right) \qquad \mathbf{q}^{\mathbf{H}_l} \in \mathbf{w} \otimes (\mathbf{x}_c - \mathbf{x}_p)^\perp \tag{32}$$

5.3.2 Simplex-simplex length analysis. With the point-simplex case in hand, the simplex-simplex eigenanalysis follows. First, set $\mathbf{x}_c \in \mathbb{R}^n$ to a weighted combination of s points $\{\mathbf{r}_0, ..., \mathbf{r}_{s-1}\}$ to obtain $\mathbf{x}_c = \sum_{i=0}^{s-1} \beta_i \mathbf{r}_i$. Next, modify $\mathbf{v}$ and $\mathbf{w}$ from Eqn. 27:

$$\mathbf{v} = \begin{bmatrix} \mathbf{r}_0 & \ldots & \mathbf{r}_{s-1} & \mathbf{p}_0 & \ldots & \mathbf{p}_{k-1} \end{bmatrix}^\top \tag{33}$$

$$\mathbf{w} = \begin{bmatrix} \beta_0 & \ldots & \beta_{s-1} & -\alpha_0 & \ldots & -\alpha_{k-1} \end{bmatrix}^\top \tag{34}$$

With $l_u = \|\mathbf{x}_c - \mathbf{x}_p\|$, the analysis in §5.3.1 holds with these variables changed, and retrieves the point-simplex case when $\beta_0 = s = 1$.

5.3.3 Collision Energy Eigensystems. Collision energies are primarily functions of unsigned length $\psi(l_u)$. By applying the chain rule, the penalty derivatives take the form

$$\frac{\partial \psi}{\partial \mathbf{v}} = \frac{\partial^2 \psi}{\partial l_u^2} \mathbf{g}_l \qquad \frac{\partial^2 \psi}{\partial \mathbf{v}^2} = \frac{\partial^2 \psi}{\partial l_u^2} \mathbf{g}_l \mathbf{g}_l^\top + \frac{\partial \psi}{\partial l_u} \mathbf{H}_l \tag{35}$$

Since $\mathbf{g}_l$ is within the null space of $\mathbf{H}_l$ and is orthogonal to its non-zero eigenpairs, the eigensystem for $\frac{\partial^2 \psi}{\partial \mathbf{v}^2}$ becomes:

$$\lambda_0^{\psi} = \frac{\partial^2 \psi}{\partial l_u^2} \left(1 + \sum_{k=0}^{n-1} \alpha_k^2 \right) \qquad \mathbf{q}_0^{\psi} = \mathbf{g}_l \tag{36}$$

$$\lambda_i^{\psi} = \frac{\partial \psi}{\partial l_u} \frac{1}{l_u} \left(1 + \sum_{k=0}^{n-1} \alpha_k^2 \right) \qquad \mathbf{q}_i^{\psi} \in \mathbf{w} \otimes (\mathbf{x}_c - \mathbf{x}_p)^\perp \tag{37}$$

The first eigenpair is a compression mode parallel to the penalty direction, and the other modes represent orthogonal buckling. To filter the Hessian, we check the signs of $\partial\psi/\partial I_u$ and $\partial^2\psi/\partial I_u^2$ and take absolute values as needed [Chen et al. 2024], avoiding any numerical eigensolving.

6 PERFORMANCE AND RESULTS

To visualize 4D meshes, we used the method of Chu et al. [2009] to maintain consistent surface normals and vertex/face ordering; their slicing method for composing surface tetrahedra is also the primary mode of visualization, and generates triangular meshes from surface tetrahedra. We used Tetgen [Si 2015] to generate tetrahedral meshes and used extrusion/filling from section 3.2 to create simulation-ready pentachoral meshes. For videos, see the supplementary materials.

6.1 Results

6.1.1 Rotating Bunny. Rotations in 3D are about an axis, but general rotations in n-D are better characterized by the two-dimensional plane they act on. In 4D, a body can deform under two completely independent rotations. We demonstrate this on a deformable bunny prism by kinematically rotating its hyper-spherical core, first with a xz rotation, then with a double rotation in yz and wx (Fig. 4).

6.1.2 Armadillo Hugs. To demonstrate the expanded range of motion in 4D, we collide two armadillo prisms. Both move along the y axis, while one rotates about the wz plane. Their contacts produce a variety of deformations that phase in and out of the 3D slice we visualize. Rotating the scene is another way to understand the interaction. A 90-degree rotation in the wx plane makes the prism-like geometry more apparent and the constant rotation in the gold armadillo more visible (Fig. 1).

6.1.3 Hyper-Dimensional Twisting. To further visualize 4D deformation, we twist several hyper-dimensional octopus prisms. Twisting two opposite ends of a prism along non-visible values of w, the 4D component only becomes visible as warping artifacts along the extremities of motion. Otherwise, objects appear to deform under an unseen influence (Fig. 5). Twisting so that vertices within the slice are forced to leave yields more effects as connected elements deform to follow along (Fig. 6).

6.1.4 4D Noodles. 3D intuitions on space and proportionality can misguide expectations when viewing 3D slices in 4D space. We drop several deforming cylinders into a four-dimensional half sphere, and observe that their deflection into the fourth dimension allows for much less space to be filled than expected (Fig. 8). Indeed, five noodles of length 20 and radius 1 only fill around 13% of the hyperbowl of radius 6, whereas in three dimensions they would fill around 69% of a 3D bowl's total volume.

6.1.5 Cantilevers. Keeping one end of a 4D cantilever constrained, the beam becomes stiffer under gravity as we increase its Young's modulus. Capturing a 3D slice in the middle of the 4D prism yields no noticeable effects. However, when choosing a slice near the prism edge, we can see sections of the beam phase in and out (Fig. 7).

6.2 Performance

Collision processing dominated the runtime for the majority of the scenarios, even after implementing AABB trees and basic multithreading. Force and Hessian assembly times similarly increase with the number of contacts. Collision processing should dominate more as the dimension increases. The $\lceil n/2 \rceil$ collision cases means that, e.g. in 5D, point-pentachoron, edge-tetrahedron, and triangle-triangle pairs need to be queried. Detailed timings for each scene are available in table 1 of the supplement.

7 CONCLUSIONS AND FUTURE WORK

7.1 Limitations

Our extrude and fill approach to meshes generates viable 4D meshes, but also a specific look. Alternate approaches like higher-order Delaunay [Aganj et al. 2007] could yield richer shapes. The "curse of dimensionality" appears for pentachoral meshes, where a large number of points are needed to obtain a high-resolution mesh, but most of the points do not appear in the final visualization. One solution might be to develop a surface-only 4D boundary-element approach [Sugimoto et al. 2022] for deformation.

We also did not investigate specially tailored strategies for collision detection or system solves that leverage the 4D structure of the simulations, so additional efficiencies are possible there.

7.2 Future Work

The deformation invariants arise as coefficients of the characteristic polynomial of $\mathbf{F}$, so higher order invariants will appear at higher dimensions. We found the existing invariants sufficed for SNH in 4D, but leave the characterization of new invariants to future work.

Finally, in the same way that the deformation equations are dimension-agnostic, the Navier-Stokes equations could also be extended to 4D. Creating a 4D fluid simulation and coupling it to our simulation is another avenue for creating previously unseen visuals.

ACKNOWLEDGMENTS

This work was supported by The Teng and Han Family Fund and NSF IIS-2132280. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

Ehsan Aganj, Jean-Philippe Pons, Florent Ségonne, and Renaud Keriven. 2007. Spatio-temporal shape from silhouette using four-dimensional delaunay meshing. In *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 1–8.

Marc Alexa, Daniel Cohen-Or, and David Levin. 2000. As-rigid-as-possible shape interpolation. In *Proceedings of SIGGRAPH (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., USA, 157–164.

Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. 2005. Variational Tetrahedral Meshing. *ACM Transactions on Graphics* (2005).

Sheldon Andrews, Kenny Erleben, and Zachary Ferguson. 2022. Contact and friction simulation for computer graphics. In *ACM SIGGRAPH Courses*. 1–172.

David Baraff and Andrew Witkin. 1998. Large steps in cloth simulation. In *Proceedings of SIGGRAPH*. 43–54. https://doi.org/10.1145/280814.280821

Javier Bonet, Antonio J Gil, and Richard D Wood. 2021. *Nonlinear solid mechanics for finite element analysis: dynamics*. Cambridge University Press.

Paul Borrel and Dominique Bechmann. 1991. Deformation of n-dimensional objects. In *Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications (SMA '91)*. Association for Computing Machinery, New York, NY, USA, 351–369. https://doi.org/10.1145/112515.112564

Table 1. Performance of 4D simulations. Symbols ϵ and k_c are distance and spring constants for point-tetrahedron and edge-trangle penalty energies. Timings are per frame in **minutes:seconds**, unless otherwise indicated. All simulations ran on 2.9GHz Intel Xeons.

Fig.	Vertices	Pentachorons	Surface Tets/Tris/Edges	Δt(s)	ϵ	k_c	Cores	Timing
4 - *Rotating Bunny*	7,235	76,176	39,618/79,236/46,280	1/30	2e-2	1000	16	6.27s
1 - *Armadillo Hugs*	18,819	205,920	103,908/207,816/121,388	1/200	8e-2	3000	12	24.5s
5a - *Alien Twisting*	23,551	245,040	132,252/264,504/154,552	1/150	5e-2	3000	12	45.1s
5b - *Octopus Twisting*	24,957	245,184	141,276/282,552/165,204	1/100	4e-2	3000	12	01:25
6 - *Thin Wringing*	26,100	218,488	147,442/294,884/172,579	1/150	3e-2	3000	12	46.9s
6 - *Thick Wringing*	28,212	280,260	157,146/314,292/183,774	1/150	4e-2	3000	12	53.8s
7 - *Cantilevers*	17,589	262,144	66,560/113,120/77,800	1/30	2e-2	3000	12	14-15s
8 - *4D Noodles*	33,415	256,000	195,200/390,400/228,420	1/150	8e-2	3000	12	49.1s

Fig. 4. *Rotating Bunny*. Kinematically constraining a hyperspherical core of an extruded Stanford bunny, we rotate it once in the xz plane (first two stills) and do a double rotation in the yz and wx planes (remaining stills).

Marc Ten Bosch. 2020. N-dimensional rigid body dynamics. *ACM Transactions on Graphics* 39, 4 (Aug. 2020), 55:55:1–55:55:6. https://doi.org/10.1145/3386569.3392483

Jan Brandts, Sergey Korotov, and Michal Křížek. 2007. Simplicial finite elements in higher dimensions. *Applications of Mathematics* 52, 3 (2007), 251–265.

Philip Claude Caplan, Robert Haimes, David L Darmofal, and Marshall C Galbraith. 2020. Four-dimensional anisotropic mesh adaptation. *Computer-Aided Design* 129 (2020), 102915.

Marco Cavallo. 2021. Higher dimensional graphics: Conceiving worlds in four spatial dimensions and beyond. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 51–63.

Honglin Chen, Hsueh-Ti Derek Liu, David I.W. Levin, Changxi Zheng, and Alec Jacobson. 2024. Stabler Neo-Hookean Simulation: Absolute Eigenvalue Filtering for Projected Newton. In *ACM SIGGRAPH Conference Papers*. New York, NY, USA, 1–10.

Alan Chu, Chi-Wing Fu, Andrew Hanson, and Pheng-Ann Heng. 2009. GL4D: A GPU-based Architecture for Interactive 4D Visualization. *IEEE Transactions on Visualization and Computer Graphics* 15, 6 (Nov. 2009).

CodeParade. 2023. Making Models in 4 Dimensions - 4D Golf Devlog #6. https://www.youtube.com/watch?v=Ir8oft_qAMQ

Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H Barr. 2001. Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of SIGGRAPH*. 31–36.

Stefan Gottschalk, Ming C Lin, and Dinesh Manocha. 1996. OBBTree: A hierarchical structure for rapid interference detection. In *Proceedings of SIGGRAPH*. 171–180.

Andrew J Hanson. 2005. Visualizing quaternions. In *ACM SIGGRAPH Courses*. 1–es.

Andrew J Hanson and Robert A Cross. 1993. Interactive visualization methods for four dimensions. In *Proceedings Visualization'93*. IEEE, 196–203.

John C Hart, Daniel J Sandin, and Louis H Kauffman. 1989. Ray tracing deterministic 3-D fractals. In *Proceedings of SIGGRAPH*. 289–296.

Linda Dalrymple Henderson. 2018. *The fourth dimension and non-Euclidean geometry in modern art*. MIT Press.

William L Hibbard, John Anderson, Ian Foster, Brian E Paul, Robert Jacob, Chad Schafer, and Mary K Tyree. 1996. Exploring coupled atmosphere-ocean models using Vis5D. *The International Journal of Supercomputer Applications and High Performance Computing* 10, 2-3 (1996), 211–222.

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral meshing in the wild. *ACM Trans. Graph.* 37, 4 (2018), 60.

Kemeng Huang, Floyd M Chitalu, Huancheng Lin, and Taku Komura. 2024. GIPC: Fast and stable Gauss-Newton optimization of IPC barrier energy. *ACM Transactions on Graphics* 43, 2 (2024), 1–18.

Dariusz Jamroz. 2009. Multidimensional labyrinth–multidimensional virtual reality. In *man-machine interactions*. Springer, 445–450.

Seung-Wook Kim, Jaehyung Doh, and Junghyun Han. 2022. Modeling and rendering non-euclidean spaces approximated with concatenated polytopes. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–13.

Theodore Kim, Fernando De Goes, and Hayley Iben. 2019. Anisotropic elasticity for inversion-safety and element rehabilitation. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–15.

Theodore Kim and David Eberle. 2022. Dynamic deformables: implementation and production practicalities (now with code!). In *ACM SIGGRAPH Courses*. 1–259.

Huancheng Lin, Floyd M Chitalu, and Taku Komura. 2024. Analytic rotation-invariant modelling of anisotropic finite elements. *ACM Transactions on Graphics* 43, 5 (2024), 1–20.

Bruno Lévy and Nicolas Bonneel. 2013. Variational Anisotropic Surface Meshing with Voronoi Parallel Linear Enumeration. In *Proceedings of the 21st International Meshing Roundtable*, Xiangmin Jiao and Jean-Christophe Weill (Eds.). Springer, Berlin, Heidelberg, 349–366. https://doi.org/10.1007/978-3-642-33573-0_21

Takanobu Miwa, Yukihito Sakai, and Shuji Hashimoto. 2015. 4-D spatial perception established through hypercube recognition tasks using interactive visualization system with 3-D screen. In *Proceedings of the ACM SIGGRAPH Symposium on Applied Perception*. 75–82.

Matthew Moore and Jane Wilhelms. 1988. Collision detection and response for computer animation. In *Proceedings of SIGGRAPH*. 289–298.

Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. 2002. Stable real-time deformations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 49–54.

Seoyong Nam, Minho Chung, Haerim Kim, Eunchae Kim, Taehyeon Kim, and Yongjae Yoo. 2024. Automatic Generation of Multimodal 4D Effects for Immersive Video Watching Experiences. In *SIGGRAPH Asia Technical Communications*. 1–4.

Alan Norton. 1982. Generation and display of geometric fractals in 3-D. *Proceedings of SIGGRAPH* 16, 3 (1982), 61–67.

Miroslav S. Petrov and Todor D. Todorov. 2018. Stable subdivision of 4D polytopes. *Numerical Algorithms* 79, 2 (Oct. 2018), 633–656. https://doi.org/10.1007/s11075-017-0454-2

Miroslav S Petrov and Todor D Todorov. 2021. Properties of the multidimensional finite elements. *Appl. Math. Comput.* 391 (2021), 125695.

Mike Seymour. 2014. Interstellar: inside the black art. *FXGuide* (2014).

Jonathan Shewchuk. 2002. What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures. *University of California at Berkeley* 2002

(a) Alien

(b) Octopus

Fig. 5. *Alien/Octopus Twisting.* By twisting the ends of an extruded prism, the middle slice appears to crush inwards on itself (top rows). Rotating the camera 90 degrees in the wx plane reveals the underlying twisting (bottom rows).

Fig. 6. *Wringing.* Kinematically constraining the tentacle tips of an extruded octopus, we first stretch it downwards (top left) and apply a 540 degree rotation in the wy plane. Depending on the extrusion thickness, different effects appear in the main slice.

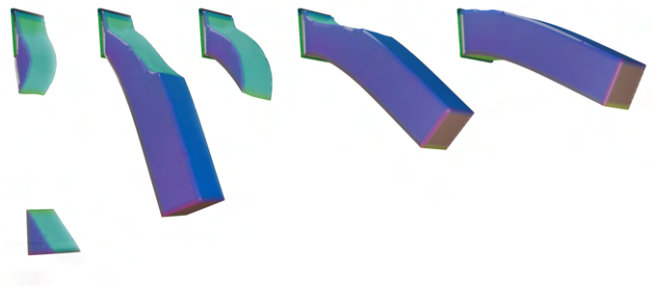

Fig. 7. *Cantilevers.* From left to right, the cantilevers sag at stiffness coefficients $E = 275, 550, 1010, 2000$, and 4000 GPa. The extra degree of freedom in the fourth spatial dimension causes parts of the beam to phase out of the slice as it settles.

(2002).

Jonathan Richard Shewchuk. 2008. General-Dimensional Constrained Delaunay and Constrained Regular Triangulations, I: Combinatorial Properties. *Discrete & Computational Geometry* 39, 1 (March 2008), 580–637. https://doi.org/10.1007/s00454-008-9060-3

Alvin Shi and Theodore Kim. 2023. A Unified Analysis of Penalty-Based Collision Energies. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6, 3 (Aug. 2023), 41:1–41:19. https://doi.org/10.1145/3606934

Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Software* 41, 2 (Feb. 2015), 11:1–11:36. https://doi.org/10.1145/2629697

Breannan Smith, Fernando De Goes, and Theodore Kim. 2018. Stable Neo-Hookean Flesh Simulation. *ACM Transactions on Graphics* 37, 2 (March 2018), 12:1–12:15. https://doi.org/10.1145/3180491

Breannan Smith, Fernando De Goes, and Theodore Kim. 2019. Analytic eigensystems for isotropic distortion energies. *ACM Transactions on Graphics (TOG)* 38, 1 (2019), 1–15.

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Proceedings of the Symposium on Geometry processing*, Vol. 4. 109–116.

Ryusuke Sugimoto, Christopher Batty, and Toshiya Hachisuka. 2022. Surface-Only Dynamic Deformables using a Boundary Element Method. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 75–86.

Jarke J van Wijk and Robert van Liere. 1993. Hyperslice. In *Proceedings Visualization*. IEEE, 119–125.

Max von Danwitz, Patrick Antony, Fabian Key, Norbert Hosters, and Marek Behr. 2021. Four-dimensional elastically deformed simplex space-time meshes for domains with time-variant topology. *International Journal for Numerical Methods in Fluids* 93, 12 (2021), 3490–3506. https://doi.org/10.1002/fld.5042 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.5042.

Fig. 8. *Hypernoodles.* 4D noodles fall into a 4D bowl. The top row depicts the strands as they deform into the fourth dimension, while the bottom row sweeps through several slices of the settled state.

Mingchuan Wang, Benoît Panicaud, Emmanuelle Rouhaud, Richard Kerner, and Arjen Roos. 2016. Incremental constitutive models for elastoplastic materials undergoing finite deformations by using a four-dimensional formalism. *International Journal of Engineering Science* 106 (Sept. 2016), 199–219. https://doi.org/10.1016/j.ijengsci.2016.06.006

Gao-Feng Zhao. 2017. Developing a four-dimensional lattice spring model for mechanical responses of solids. *Computer Methods in Applied Mechanics and Engineering* 315 (March 2017), 881–895. https://doi.org/10.1016/j.cma.2016.11.034

Zichun Zhong, Wenping Wang, Bruno Lévy, Jing Hua, and Xiaohu Guo. 2018. Computing a high-dimensional euclidean embedding from an arbitrary smooth riemannian metric. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–16.