PHYSICS-BASED SPHERE PACKING FOR LAGRANGIAN MESH MORPHING

Anonymous authors
Paper under double-blind review

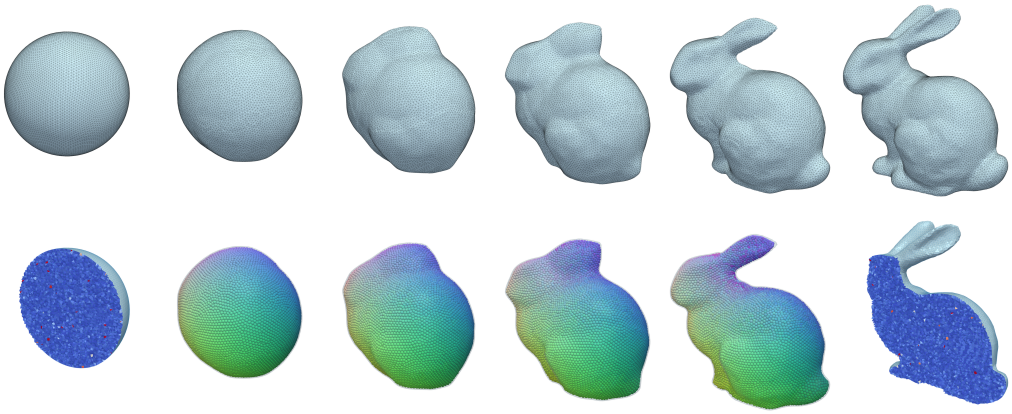


Figure 1: Sphere-to-bunny morph that keeps sphere identities throughout. Top: surface mesh per step; bottom: the spheres, coloured by starting position, settled in each surface and connected into tetrahedra; initial and target meshes cut open, coloured by element quality: regular distorted.

ABSTRACT

This paper studies tetrahedral meshes as the body representation for differentiable simulation and computational design. Both change the shape of a body throughout an optimization and need meshes that follow large shape changes while keeping their interior nodes in correspondence. Fixed-connectivity meshes degrade under large morphs, while remeshing from scratch discards node correspondence. We present JamTet, a physics-based sphere-packing framework for volumetric meshing and morphing. We contribute (i) a GPU-parallel mesher combining octree-hierarchical packing with constrained Delaunay tetrahedralization, producing lower element-volume spread than TetGen and fTetWild; (ii) Lagrangian mesh morphing that preserves interior-node identities by re-equilibrating the same spheres within changing shapes and rebuilding the boundary and connectivity, remaining inversion-free where fixed-connectivity and TetSphere meshes invert; and (iii) a differentiable GPU simulator in JAX, with mass–spring edges and a volumetric Neo-Hookean term, integrated with mesh morphing in a design pipeline. In soft-robot morphology design experiments, interior-node gradients improve swimming fitness by 0.73–1.07 over a matched surface-only variant, while voxelized versions of the same designs yield 32–63% lower fitness. These results establish sphere packing as a practical volumetric mesh representation for gradient-based shape optimization. Code and media: `https://anonymous.4open.science/r/jamtet/`.

1 INTRODUCTION

Gradient-based shape optimization with differentiable physics changes the geometry of a body step by step, and the volumetric mesh used by the simulator has to conform to each new boundary with well-shaped tetrahedra. If its interior nodes also keep their identities, values stored on those nodes

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

carry over from one shape to the next. TetSphere splatting (Guo et al., 2025) keeps node identity with a Lagrangian volumetric mesh, a union of tetrahedralized spheres whose nodes move with the shape, and discourages inverted elements with a soft penalty; in our morph experiments, some of its tetrahedra still invert. Tetrahedral meshers such as TetGen (Si, 2015) and fTetWild (Hu et al., 2020a) conform to a given boundary and control element quality, and each call builds a new mesh with new node indices. A mesh that changes shape thus has two standard options: fixed connectivity keeps node identity while its elements distort, inverting 4.1% of the tetrahedra by the end of a sphere-to-bunny morph, and remeshing from scratch restores element quality while assigning new node indices (Section 4.2). We target a conforming tetrahedral mesh whose interior nodes persist and whose elements stay well shaped through large shape changes.

Soft-robot morphology design is one application that needs such meshes. Evolutionary design grows soft bodies from voxels (Cheney et al., 2013; Bhatia et al., 2021), and recent systems design voxel robots with learned controllers (Li et al., 2025) or pretrain them in differentiable simulation (Strgar & Kriegman, 2026). ChainQueen (Hu et al., 2019) and DiffTaichi (Hu et al., 2020b) simulate soft robots with differentiable particle and mass–spring models. Voxel bodies keep one connectivity and a boundary of cube faces, and particle bodies have no element connectivity (Appendix A).

JamTet builds the mesh from physical spheres (Figure 2a). Given a surface, spheres flow inside it like a liquid and then jam into a dense packing (O'Hern et al., 2003); their centres become the interior nodes, and constrained Delaunay tetrahedralization (CDT) connects them with the nodes of the re-triangulated surface. To morph, the same spheres re-settle inside each new surface and CDT rebuilds the connectivity, so every interior node keeps its identity (Figure 2b). A differentiable GPU simulator in JAX runs on these meshes, and a design loop turns its gradients into new shapes through the morph step, so every design keeps the same interior spheres.

We make three contributions. (i) A sphere-packing mesher with GPU packing, an octree schedule and CDT connectivity, whose meshes match or exceed the element quality of TetGen and fTetWild with more uniform element volumes (Section 4.1). (ii) Identity-preserving morphing: along a sphere-to-bunny morph, the mesh keeps every interior node identity and has no inverted tetrahedron at any step (Section 4.2). (iii) A differentiable GPU simulator with mass–spring edges and a volumetric Neo-Hookean term, coupled with the morph step in a morphology-design pipeline. For soft swimmers, gradients on interior nodes raise fitness by 0.73–1.07 over a matched surface-only variant in all four swimming conditions, and voxel versions of the designs reach 32–63% lower fitness than the smooth designs (Section 4.3).

Section 2 reviews related work, Section 3 describes the method, Section 4 reports the experiments, and Section 5 discusses limitations and concludes.

2 Related Work

Tetrahedral meshing. Delaunay refinement (Shewchuk, 1998) and constrained Delaunay tetrahedralization, as in TetGen (Si, 2015), recover a given boundary and insert Steiner points to bound element quality. fTetWild (Hu et al., 2020a) meshes imperfect input within a tolerance envelope and then optimizes element quality. Among particle methods, bubble packing relaxes bubbles with simulated forces before Delaunay triangulation (Shimada & Gossard, 1995), ellipsoidal bubble packing connects the nodes by an advancing front for anisotropic meshes (Yamakawa & Shimada, 2000), later sphere packings vary the element size or pack faster (Lo & Wang, 2005; Liu et al., 2008; Qi et al., 2014), and DistMesh balances spring forces inside a signed distance field (Persson & Strang, 2004). JamTet builds on sphere packing with a two-stage jamming schedule, an octree and GPU execution, and it carries one packing across shape changes.

Deforming volumetric representations. The material point method (MPM) (Sulsky et al., 1994; Hu et al., 2018) carries a deforming body on particles and solves its motion on a background grid, and ChainQueen (Hu et al., 2019) differentiates it for soft-robot design. A tetrahedral mesh gives the body an explicit boundary and elements on which finite-element analysis runs directly; Appendix F uses our meshes for compliance design with linear finite elements. DefTet (Gao et al., 2020) deforms a fixed tetrahedral grid and predicts per-cell occupancy. TetSphere splatting (Guo et al., 2025) represents a shape as a union of deformable tetrahedralized spheres whose nodes persist during optimization, and it is the closest prior representation. DMTet (Shen et al., 2021), Flexi-

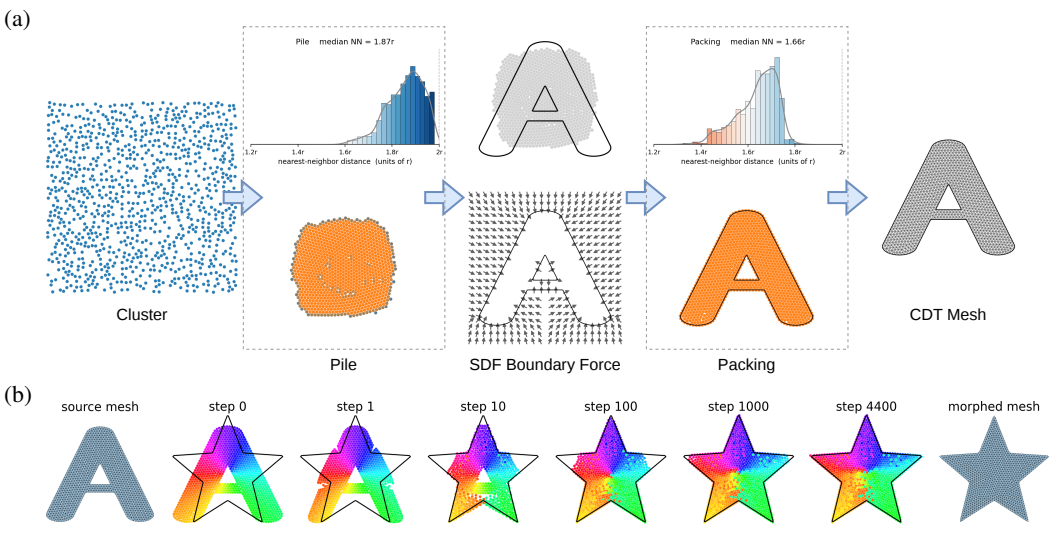


Figure 2: Sphere packing mesh in 2D. (a) Mesh generation, left to right: a loose cluster settles into a pile, a boundary force pushes it into the target shape, relaxation gives a tight packing (insets: neighbour distances), and constrained Delaunay triangulation gives the mesh. (b) Morphing: a packed letter "A" morphs into a star, and the balls, coloured by starting position (hue: angle; brightness: radius), keep their identities.

Cubes (Shen et al., 2023) and TetWeave (Binninger et al., 2025) extract surfaces from fields stored on deformable tetrahedral, cubical or Delaunay background grids, which TetWeave rebuilds from optimized points every few steps. Level-set methods morph solids by evolving an implicit function (Breen & Whitaker, 2001), and our morphs follow such a path.

Differentiable simulation and morphology design. ChainQueen (Hu et al., 2019), DiffTaichi (Hu et al., 2020b), PlasticineLab (Huang et al., 2021) and DiffPD (Du et al., 2021) differentiate soft-body rollouts with respect to states and parameters. DiffAqua (Ma et al., 2021) optimizes soft swimmer shapes with such gradients. ELDiR (Strgar et al., 2024) evolves robot bodies while training their neural controllers through a differentiable mass–spring simulator on the GPU. Appendix A compares the body representations of these systems.

3 Method

Section 3.1 packs spheres inside a surface and connects them into tetrahedra, Section 3.2 accelerates the packing, Section 3.3 carries the spheres through a sequence of shapes, and Section 3.4 simulates the meshes and turns gradients into shape updates. Throughout, Ω is the domain inside a closed surface with signed distance function ϕ ($\phi < 0$ inside), n the number of interior spheres, s the node spacing, $\mathcal{S}$ the surface nodes, and $\mathbf{p}_i$ and $\mathbf{v}_i$ the centre and velocity of sphere i, with $\mathcal{P} = \{\mathbf{p}_i\}_{i=1}^{n}$.

3.1 Surface-first sphere packing

One length sets the resolution: the node spacing $s = 0.93\,(|\Omega|/n)^{1/3}$, which matches the median nearest-neighbour distance of our packings. Every sphere has radius $r = (3|\Omega|/4\pi n)^{1/3} \approx 0.667\,s$, so the spheres together have the volume of Ω and overlap their neighbours. Meshing runs in three steps (Figure 2a). First, the boundary is re-triangulated with isotropic triangles of edge length s (Botsch & Kobbelt, 2004), whose nodes $\mathcal{S}$ stay fixed, and ϕ is recomputed from this surface on a 128^3 grid. Second, n spheres are packed inside it. Third, CDT connects $\mathcal{S} \cup \mathcal{P}$ into tetrahedra.

Packing moves the centres under two pair forces applied in sequence. The *water* stage uses a long-range Gaussian repulsion,

$$\mathbf{f}_i^{\mathrm{w}} = k_{\mathrm{w}} \sum_{j \neq i} \exp\!\big(-\|\mathbf{p}_i - \mathbf{p}_j\|^2/s^2\big)\,\hat{\mathbf{u}}_{ij}, \tag{1}$$

where $\hat{\mathbf{u}}_{ij}$ is the unit vector from $\mathbf{p}_j$ to $\mathbf{p}_i$. The force acts at every distance up to its cutoff $2.2\,s$, so spheres keep sliding past one another and flow into empty regions. The stage runs only while the packing has a hole, that is, while some interior point deeper than s lies farther than $1.6\,s$ from every centre. The *jam* stage uses a repulsion-only contact force,

$$\mathbf{f}_i^{\mathrm{j}} = k_{\mathrm{c}} \sum_{j \neq i} \max\big(2r - \|\mathbf{p}_i - \mathbf{p}_j\|,\, 0\big)\, \hat{\mathbf{u}}_{ij}, \tag{2}$$

which acts only between overlapping spheres. Each sphere becomes caged by its neighbours, and the damped motion settles into a jammed packing; the stage stops when the mean speed falls below 10^{-4}. Both stages integrate

$$\mathbf{v}_i \leftarrow \gamma\, \mathbf{v}_i + \Delta t\, \mathbf{f}_i, \qquad \mathbf{p}_i \leftarrow \Pi\big(\mathbf{p}_i + \Delta t\, \mathbf{v}_i\big), \tag{3}$$

with unit mass, $\gamma = 0.82$, $\Delta t = 0.005$, $k_{\mathrm{w}} = 300$ and $k_{\mathrm{c}} = 500$, and each step is capped at $\|\mathbf{v}_i\|\,\Delta t \le c\,s$, with $c = 0.5$ for jam and $c = 0.12$ for water.

The projection Π keeps every centre at least s inside the surface. It moves each centre with $\phi(\mathbf{p}) > -s$ onto the level set $\phi = -s$ by five Newton steps $\mathbf{p} \leftarrow \mathbf{p} - (\phi(\mathbf{p}) + s)\,\nabla\phi/\|\nabla\phi\|$ and removes the outward normal component of its velocity. The first interior layer therefore sits one spacing below the surface, and boundary tetrahedra have the size of interior ones. At full resolution, the surface nodes $\mathcal{S}$ enter Eq. (2) as fixed spheres, so the first layer settles into the hollows above the surface triangles. TetGen (Si, 2015) then computes the CDT of $\mathcal{S} \cup \mathcal{P}$ with the surface triangles as constraints, and the vertices of the mesh are exactly these nodes. The mesh stores the surface nodes first and the interior spheres next, in a fixed order, so the i-th interior node is sphere i. The design loop adds TetGen's quality refinement (radius–edge bound 1.414), which appends helper nodes after the input nodes, so the interior block is unchanged.

3.2 Octree-hierarchical packing

Direct packing of $n = 64$k spheres runs for thousands of steps, and each step rebuilds a spatial hash to find neighbours. The octree schedule builds the packing from coarse to fine (Figure 3). Level 0 packs at most $n/64$ spheres at spacing $4s$ with the two stages of Section 3.1. Each finer level ℓ, with spacing s_ℓ, replaces every sphere by eight children at the corners of a cube of side s_ℓ centred on it. It then injects spheres at grid points farther than $0.95\,s_\ell$ from every centre, up to the level's target count, which fills thin features that no coarse sphere reached. The children start near equilibrium and move less than one spacing, so each level builds one neighbour list, refreshes it every 200 steps, and runs the jam stage on this cached list; the water stage runs only if the fill check fails. A step then costs one gather over a fixed index array.

3.3 Morphing with persistent spheres

A morph moves the same spheres through a sequence of shapes. A level-set flow (Breen & Whitaker, 2001) carries the source ϕ_0 towards the target ϕ_1,

$$\partial_\tau \phi = \phi_1\, \|\nabla\phi\|, \tag{4}$$

which grows the shape where it lies inside the target and shrinks it outside, so the surface approaches the target monotonically. We take snapshots evenly spaced in the resolved fraction of the source–target symmetric-difference volume. Each snapshot surface is extracted, smoothed (Taubin, 1995), scaled to the target volume so that the fixed set of spheres keeps spacing s, and re-triangulated. The spheres of the previous step are projected inside the new surface, the water stage carries them into new volume until the fill check passes, and the jam stage settles them against the new surface nodes. CDT then rebuilds the connectivity. The number of spheres stays fixed, so interior node i at every step is sphere i of the source packing, while the surface is re-triangulated at every step.

3.4 Differentiable simulation and design

The mesh carries a mass–spring model with a volumetric Neo-Hookean term. Every tetrahedron edge $e = (i, j)$ is a damped spring with rest length L_e, and every tetrahedron τ with rest volume V_τ adds a Neo-Hookean volume energy in its volume ratio J_τ:

$$E(\mathbf{x}) = \sum_e \tfrac{1}{2}\, k_s\big(\|\mathbf{x}_i - \mathbf{x}_j\| - a_e(t)\, L_e\big)^2 + \sum_\tau \mu\, V_\tau\big(J_\tau - 1 - \ln J_\tau\big). \tag{5}$$

The tetrahedral mesh defines this model directly: its nodes carry the mass, its edges form the spring network, and the jammed packing gives edges of similar length. A spring network has no measure of volume, so under large deformation a tetrahedron can collapse or invert. The well-shaped tetrahedra of our meshes therefore also carry the Neo-Hookean volume term, whose energy rises steeply as J_τ approaches zero; it resists collapse and inversion and approximates the near incompressibility of soft tissue. The activation $a_e(t)$ scales the rest length of muscle edges and equals one for passive edges. In the implementation, J_τ is clamped at 0.01, and μ is scaled down for tetrahedra whose rest volume lies below 10% of the median. Semi-implicit Euler integrates the nodes, which carry equal masses. For swimming, water acts through surface forces (Min et al., 2019; Ma et al., 2021). We apply a linear drag $-A\,(c_n \mathbf{v}_n + c_t \mathbf{v}_t)$ to each boundary triangle with area A, where $\mathbf{v}_n$ and $\mathbf{v}_t$ are the normal and tangential parts of its velocity and $c_n = 6\,c_t$. The whole simulator is written in JAX (Bradbury et al., 2018) and runs on the GPU as one compiled scan over time steps, fast enough to iterate on designs: a 6 s rollout takes 0.4–2.3 s from 1k to 64k spheres (Appendix C). Automatic differentiation gives the gradient of a rollout objective with respect to every node's rest position in 3.6–6.3 s at 1k and 8k, so the morphology optimization uses simulator gradients directly. The backward pass recomputes each block of steps from a checkpoint (500 steps at 1k spheres, 125 at 8k), so memory grows with the number of blocks, and at each step it clips the norm of the state cotangent to 10^3 for numerical stability.

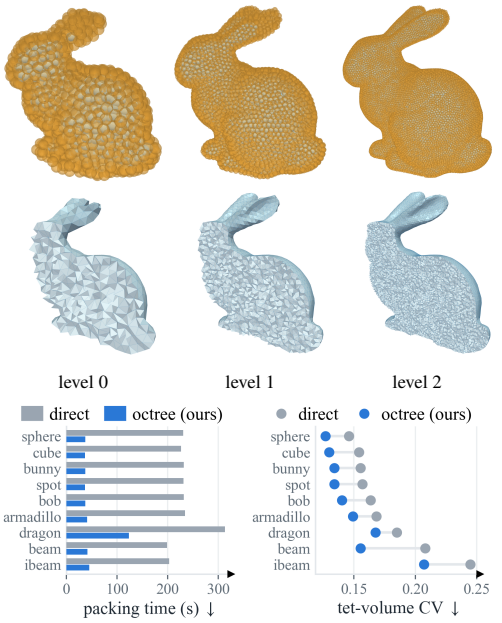


Figure 3: Octree packing on the bunny. Top: three levels (each ball split into 8), packing and cut mesh; bottom: packing time and coefficient of variation (CV) of tetrahedron volumes at 64k nodes per shape, octree vs. direct.

Table 1: Static mesh quality at 64k interior spheres, per shape; measures defined in Section 4.1; no inverted elements for any method; all nine shapes: Table A2.[1]

Method	Vol CV (σ/μ) ↓	AMIPS Avg±std↓	Min Dih deg↑	Aspect Avg±std↑	Wall (s)
Bunny					
TetGen	0.296	3.81 ± 0.88	48.6 ± 12.6	0.573 ± 0.133	0.5
fTetWild	0.540	3.33 ± 0.26	56.0 ± 7.3	0.718 ± 0.137	77.2
Ours	0.134	3.33 ± 0.90	57.0 ± 9.2	0.733 ± 0.128	36.9
Spot					
TetGen	0.296	3.80 ± 0.88	48.7 ± 12.6	0.574 ± 0.133	0.5
fTetWild	0.332	3.49 ± 0.30	53.1 ± 8.6	0.621 ± 0.112	50.3
Ours	0.134	3.33 ± 0.87	56.9 ± 9.2	0.732 ± 0.127	36.4

Design loop. Each design iteration turns the fitness of a rollout into a new shape in four steps. (1) The simulator computes the gradient of the fitness with respect to the rest position of every node, averaged over eight perturbed rollouts. (2) The node gradients are smoothed on a grid and transferred to the design parameters, the displacement vectors of a free-form deformation (FFD) lattice (Sederberg & Parry, 1986) that warps the level set of the body. (3) A heavy-ball step with momentum 0.9 updates these displacements. (4) The morph step of Section 3.3 re-packs the spheres inside the warped shape and rebuilds the tetrahedral mesh with hybrid CDT. Gradients reach the node positions through the simulator, while the morph step maps each level set to a valid mesh and passes no gradient.

4 Experiments

We evaluate JamTet in three settings: static meshing of nine shapes against TetGen and fTetWild (Section 4.1), morphing against fixed connectivity, fTetWild and TetSphere (Section 4.2), and soft-robot morphology design with the differentiable simulator (Section 4.3). The appendix gives the full tables and settings (Appendices B–E) and a finite-element design study (Appendix F).

[1] Shading in all tables: best , second best .

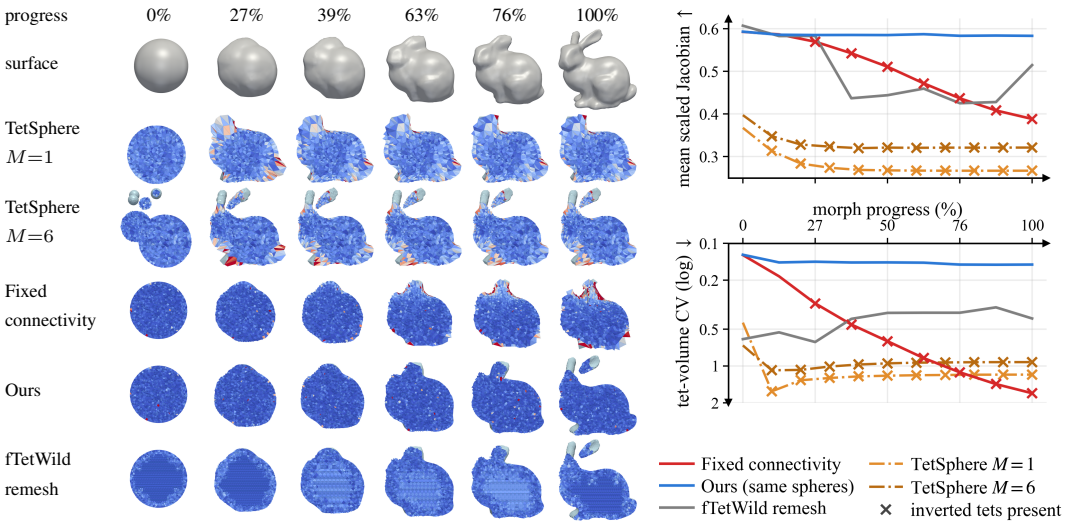


Figure 4: Sphere-to-bunny morph quality, ≈10k spheres. Left: surface per step, shared by all but TetSphere; cut views coloured by AMIPS energy, from blue (3, regular) to deep red (8 or more). Right: mean scaled Jacobian and volume CV per step; crosses: inverted elements present.

4.1 Static meshing

We mesh nine shapes, from a sphere, a cube and two beams to the bunny, Spot, Bob, Armadillo and the dragon, with $n = 64$k interior spheres. TetGen (switches `pYq1.414a`) and fTetWild mesh the same shapes, with the maximum tetrahedron volume or the edge length tuned to our node count. Our packing runs on one GPU; our surface triangulation and CDT, TetGen and fTetWild run on the CPU. Table 1 reports five measures: *Vol CV*, the coefficient of variation of tetrahedron volumes; *AMIPS*, the mean AMIPS energy (Fu et al., 2015), 3 for a regular tetrahedron; *Min Dih*, the smallest dihedral angle of each tetrahedron averaged over the mesh, 70.5° when regular; *Aspect*, a normalized aspect ratio, 1 when regular; and *Wall*, the time of the whole meshing pipeline.

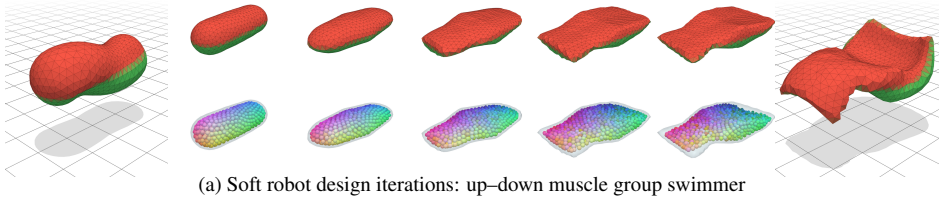
On the bunny and Spot, our meshes rank first in all four quality measures, with AMIPS tied with fTetWild on the bunny, and second in wall time (Table 1). Their Vol CV is 45% of that of TetGen and 25–40% of that of fTetWild. No mesher produces an inverted element. Over all nine shapes, our meshes have the lowest Vol CV on every shape (Appendix B).

The low volume spread is consistent with the packing, whose jam stage leaves neighbouring centres at nearly equal distances. The octree schedule (Figure 3) packs 2.5–6.3× faster than direct packing on the nine shapes and gives a lower Vol CV on every shape.

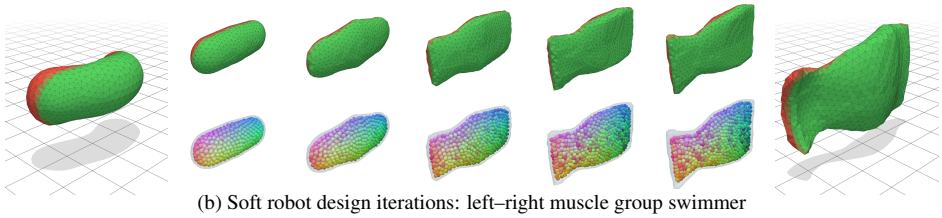
4.2 Identity-preserving morphing

We morph a sphere into the bunny along the level-set path of Section 3.3 in nine steps, evenly spaced in the resolved fraction of the symmetric-difference volume, with ϕ on a 256^3 grid and 9,941 interior spheres. Ours, fixed connectivity and fTetWild take the same nine surfaces as input and differ in how they treat the interior. Ours re-packs the same spheres and rebuilds the connectivity at each step. Fixed connectivity keeps the step-0 mesh, projects its surface nodes onto each new level set, and moves the interior nodes by harmonic extension of the surface motion. fTetWild meshes each step from scratch with its edge length set for our node count. TetSphere (Guo et al., 2025) with one ($M=1$) or six ($M=6$) spheres runs its own 11-step morph of the same pair at about 10k nodes.

Ours has no inverted tetrahedron at any step, and its Vol CV stays within 0.124–0.149 (Table 2). Fixed connectivity starts from the same mesh and degrades step by step: inverted tetrahedra appear at step 2 and reach 4.1% at the bunny. fTetWild has no inverted tetrahedron and three times our volume spread, and TetSphere inverts tetrahedra at every step after the first. Re-packing restores the sphere spacing at every step, while fixed connectivity stretches the source tetrahedra over the new volume. A fresh octree packing at every step, which discards identity, reaches the same Vol CV as



(a) Soft robot design iterations: up–down muscle group swimmer



(b) Soft robot design iterations: left–right muscle group swimmer

Figure 5: Morphology design driven by gradients on surface and interior nodes, 1k nodes: starting capsule mid-stroke, five example iterations, and best design mid-stroke; muscle groups red and green; below each design: its interior spheres, coloured by starting position.

Table 2: Morph quality, mean±sd over the morph steps; rows as in Figure 4; Identity: interior node identities kept; †own morph protocol (Section 4.2).

Method	Identity	Scaled $J\uparrow$	% Inv $\downarrow$	Min Dih (°) $\uparrow$	AMIPS $\downarrow$	Vol CV $\downarrow$
TetSphere $M{=}1^{\dagger}$	yes	0.283 ± 0.030	4.70 ± 1.85	38.9 ± 2.9	6.41 ± 0.95	1.177 ± 0.261
TetSphere $M{=}6^{\dagger}$	yes	0.331 ± 0.022	0.44 ± 0.45	42.1 ± 1.7	4.51 ± 0.30	0.950 ± 0.101
Fixed connectivity	yes	0.501 ± 0.078	1.11 ± 1.59	52.4 ± 4.4	4.72 ± 1.94	0.753 ± 0.552
Ours	yes	0.586 ± 0.003	0.00 ± 0.00	56.5 ± 0.3	3.36 ± 0.02	0.143 ± 0.008
fTetWild remesh	no	0.498 ± 0.075	0.00 ± 0.00	54.0 ± 2.2	3.46 ± 0.13	0.449 ± 0.113

Table 3: Fitness by method, task and mesh size: forward displacement of the centre of mass, averaged over the second half of the rollout, in simulation units; mean±sd over three seeds.

environment	scale	ours	ours-surface	remesh-tetgen	remesh-ftw	fixed-conn
swim, up–down	1k	2.469 ± 0.123	1.395 ± 0.019	1.300 ± 0.009	1.254 ± 0.013	0.535 ± 0.210
swim, lateral	1k	2.345 ± 0.031	1.413 ± 0.027	1.283 ± 0.031	1.226 ± 0.023	0.618 ± 0.239
loco, up–down	1k	1.831 ± 0.087	1.845 ± 0.288	1.878 ± 0.407	1.925 ± 0.543	1.544 ± 0.211
swim, up–down	8k	2.488 ± 0.035	1.761 ± 0.010	1.700 ± 0.007	1.687 ± 0.016	0.600 ± 0.002
swim, lateral	8k	2.617 ± 0.079	1.781 ± 0.012	1.682 ± 0.009	1.696 ± 0.026	0.601 ± 0.001
loco, up–down	8k	1.837 ± 0.148	1.977 ± 0.673	1.775 ± 0.766	2.425 ± 0.460	1.542 ± 0.613

ours, 0.144 against 0.143, while our interior nodes keep their identities at every step: the sphere colours in Figures 1 and 2b follow each sphere from the source to the target. The same procedure carries 64,000 spheres through the morph of Figure 1 without an inverted tetrahedron.

4.3 Soft robot design study

We use the simulator and the morph step to design soft robots (Figure 5). A robot is a soft body whose two muscle groups contract in a travelling wave. Every run starts from the same capsule, 1.91 long in simulation units, meshed with about 1k or 8k interior spheres. We test swimming with an up–down and with a left–right muscle split, without gravity, and walking on frictional ground with the up–down split. Fitness is the forward displacement of the centre of mass, averaged over the second half of the rollout; Appendix D lists the settings. Each design iteration runs the mass–spring and Neo-Hookean simulator of Section 3.4 for 6 s, updates the body shape and morphs the mesh into it. Because the simulator is written in JAX, it is differentiable and a rollout with its gradient takes 3.6–6.3 s on one GPU, so the shape update follows simulator gradients directly.

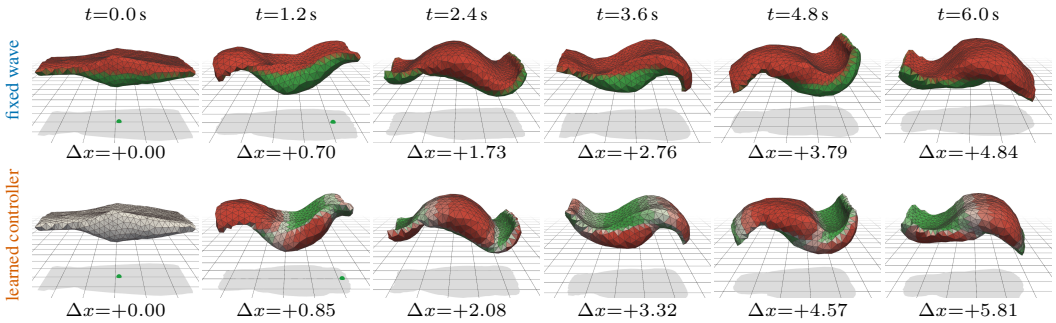


Figure 6: Fixed-wave and learned-controller swimmers: one design at the same time points. Top: fixed travelling wave, coloured by muscle group; bottom: learned controller, coloured by muscle activation; Δx: forward displacement at time t.

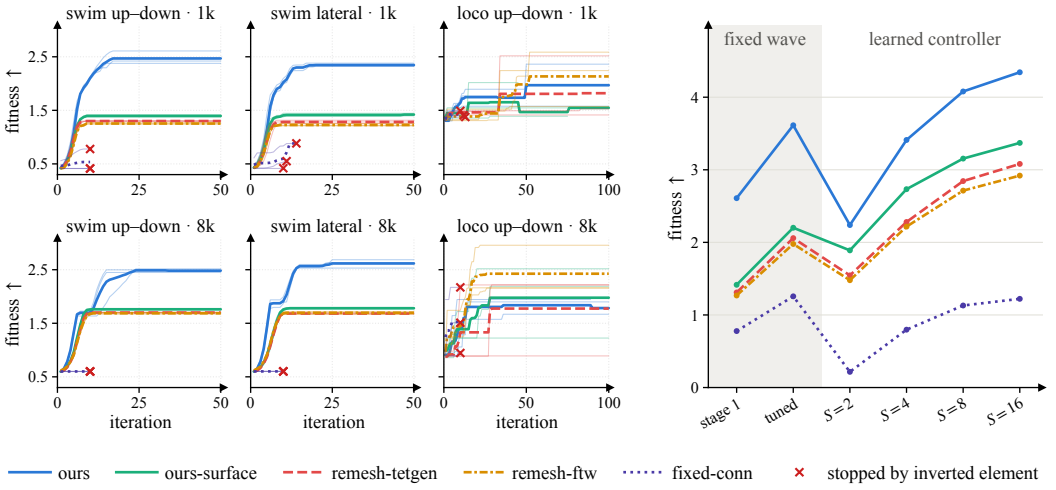


Figure 7: Shape-optimization curves and controller tests. Left: fitness per design iteration at 1k (top) and 8k nodes (bottom), walking at 1k from a separate 100-iteration rerun; thin lines: single runs; thick lines: mean of active runs. Right: the best 1k up–down designs at six test points: design-stage wave, tuned wave, and learned controllers of size S=2–16.

Five methods share the start shape, the simulator, the gait and the budget. *Ours* takes the gradient over all nodes, interior included, and *ours-surface* only over the surface nodes. *Remesh-TetGen* and *remesh-fTetWild* also use surface-node gradients and rebuild the volume from scratch at every iteration, at the node density of ours. *Fixed connectivity* moves all nodes along a smooth, volume-preserving displacement field built from the node gradients and keeps the initial tetrahedra. Each method runs three seeds in each of the six environment–scale conditions under one protocol, and a run is scored by the best fitness it reaches in 50 iterations, or 100 for walking at 8k. In all four swimming conditions, ours reaches the highest fitness and exceeds ours-surface by 0.73–1.07, and the seed ranges of the two methods are disjoint (Table 3). Ours and ours-surface share every step except the gradient support, so this margin measures what the interior gradients contribute. The remeshing baselines finish close to ours-surface, and fixed connectivity stops near iteration 10, when an element inverts. Section 5 discusses walking.

Both swimmers turn the capsule into a plate that faces the direction of bending (Figure 5). In the drag model, a face moving along its normal meets six times the resistance of a face sliding along itself, so broad faces that sweep along their normals produce thrust, and the bending that moves them depends on how material is distributed through the whole body. Gradients on interior nodes measure how moving material inside the body changes the fitness; surface-node gradients measure how moving the body outline changes it.

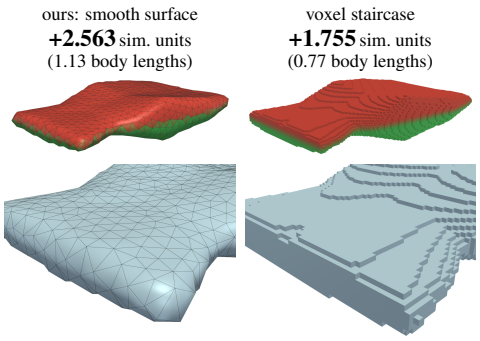


Figure 8: Mesh and voxel design representations: one swimming design with a smooth surface (left) and its voxel version on the 128^3 design grid (right), with their fitness.

A design ranked under one fixed gait may lose its rank under a stronger controller, so we train learned controllers on the best up–down design of each method at 1k: a shared multilayer perceptron maps clock signals and the rest position of each muscle block to its activation and is trained through the simulator (Appendix D). Each controller starts from a least-squares fit to the travelling wave, and each test point is scored by the median fitness over three start phases. The order ours $>$ ours-surface $>$ remesh-TetGen $>$ remesh-fTetWild $>$ fixed connectivity holds at all six test points of Figure 7, and the ratio of ours to ours-surface falls from 1.84 after design to 1.29 with the largest controller. In Figure 6, the learned controller carries our design farther than the fixed wave. With the largest controller, ours reaches a fitness of 4.34, 1.20$\times$ its fitness under the tuned wave, and four of the five methods exceed their tuned-wave fitness. Played back for 6 s, the best 1k up–down design of ours travels 3.50 under the design-stage wave and 5.81 under the learned controller, against 1.69 and 3.89 for remesh-fTetWild (Figure A2). The final designs show where the interior gradients lead (Figure A1): in up–down swimming, ours flattens the capsule to a height-to-width ratio of 0.30–0.31, while the three surface-gradient methods reach 0.65–0.74; in left–right swimming, ours raises the ratio to 3.51–3.97, against 1.33–1.54; fixed connectivity keeps 0.82–1.27, close to the capsule. Interior gradients thus carry the body towards the broad plates that the drag model rewards.

To evaluate voxel versions of our swimming designs, we replace their tetrahedral body by a voxel body with the same gait and objective, at a resolution matched to our surface and on the 128^3 design grid (Figure 8). The voxel bodies reach 32–63% lower fitness than the smooth designs at both resolutions (Appendix E).

5 Discussion

Limitations and future work. (i) *Computational cost.* Our pipeline needs a GPU and meshes a shape more slowly than TetGen, and thin boundary tetrahedra shorten the stable time step of explicit integration compared with TetGen and fTetWild meshes. (ii) *Identity.* To keep the surface triangles well shaped, every step re-triangulates the surface before CDT, so surface nodes are not tracked and only interior nodes persist; in the design study the optimizer keeps its momentum on the FFD lattice, so identity serves correspondence without a direct path to fitness, and element quality varies together with identity across the baselines. (iii) *Experimental scope.* Walking remains inconclusive, because ground contact makes its gradient non-smooth, and GPU atomics make repeated gradient evaluations differ slightly (Appendix C); a few conditions differ in budget or protocol: walking at 8k, four runs cut short by CDT failures, one controller run per condition, fTetWild overshooting its node target at the first morph steps, TetSphere's own morph, and the voxel discretization; the simulator uses simulation units with equal node masses, so body mass grows with node count. In future work, a material field stored on the persistent nodes and interpolated to each new mesh would give identity a direct role in design.

Conclusion. JamTet builds tetrahedral meshes from physical sphere packings: spheres flow and jam inside a re-triangulated surface, and CDT connects them into a conforming mesh with more uniform element volumes than TetGen and fTetWild. Because the same spheres re-settle inside every new shape, interior nodes keep their identities through large shape changes while the connectivity follows the shape, and the meshes stay free of inverted elements where fixed-connectivity and TetSphere meshes invert. A differentiable GPU simulator couples these meshes with morphology design, where gradients on interior nodes raise swimming fitness by 0.73–1.07 over surface-only gradients, and our designs reach 1.5–2.7$\times$ the fitness of their voxel versions. The same meshes also support compliance design with linear finite elements (Appendix F); together, these results make sphere packing a practical volumetric mesh representation for gradient-based shape optimization.

REPRODUCIBILITY STATEMENT

Our code at `https://anonymous.4open.science/r/jamtet/` covers meshing, morphing, simulation and the design loop. Section 3 and Appendix D give the settings of packing, morphing, simulation and design; Appendix C reports the simulator's speed and gradient checks; Appendices B, E and F give the full static, voxel and finite-element results. The design study runs three seeds per method and condition. Experiments run on NVIDIA RTX 5090 GPUs under Ubuntu.

AI USE STATEMENT

The author used an LLM for language editing, formatting, and as a search tool in the literature review, in parts of this paper and its appendix. All scientific ideas, claims, decisions, and the selection and interpretation of the cited literature are those of the researcher.

REFERENCES

Jagdeep Singh Bhatia, Holly Jackson, Yunsheng Tian, Jie Xu, and Wojciech Matusik. Evolution Gym: A large-scale benchmark for evolving soft robots. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 2201–2214, 2021.

Alexandre Binninger, Ruben Wiersma, Philipp Herholz, and Olga Sorkine-Hornung. TetWeave: Isosurface extraction using on-the-fly Delaunay tetrahedral grids for gradient-based mesh optimization. *ACM Transactions on Graphics*, 44(4), 2025.

Mario Botsch and Leif Kobbelt. A remeshing approach to multiresolution modeling. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pp. 185–192, 2004.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs. `http://github.com/jax-ml/jax`, 2018.

David E. Breen and Ross T. Whitaker. A level-set approach for the metamorphosis of solid models. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):173–192, 2001.

Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 167–174, 2013.

Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. DiffPD: Differentiable projective dynamics. *ACM Transactions on Graphics*, 41(2): 13:1–13:21, 2021. doi: 10.1145/3490168.

Xiao-Ming Fu, Yang Liu, and Baining Guo. Computing locally injective mappings by advanced MIPS. *ACM Transactions on Graphics*, 34(4):71:1–71:12, 2015.

Jun Gao, Wenzheng Chen, Tommy Xiang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3D reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 9936–9947, 2020.

Minghao Guo, Bohan Wang, Kaiming He, and Wojciech Matusik. TetSphere splatting: Representing high-quality geometry with Lagrangian volumetric meshes. In *International Conference on Learning Representations (ICLR)*, 2025.

Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. Fast tetrahedral meshing in the wild. *ACM Transactions on Graphics*, 39(4):117:1–117:18, 2020a.

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics*, 37(4), 2018. doi: 10.1145/3197517.3201293.

Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B. Tenenbaum, William T. Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. ChainQueen: A real-time differentiable physical simulator for soft robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. DiffTaichi: Differentiable programming for physical simulation. In *International Conference on Learning Representations (ICLR)*, 2020b.

Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B. Tenenbaum, and Chuang Gan. PlasticineLab: A soft-body manipulation benchmark with differentiable physics. In *International Conference on Learning Representations (ICLR)*, 2021.

Muhan Li, Lingji Kong, and Sam Kriegman. Generating freeform endoskeletal robots. In *International Conference on Learning Representations (ICLR)*, 2025.

Jianfei Liu, Shuixiang Li, and Yongqiang Chen. A fast and practical method to pack spheres for mesh generation. *Acta Mechanica Sinica*, 24(4):439–447, 2008. doi: 10.1007/s10409-008-0165-y.

S. H. Lo and W. X. Wang. Generation of tetrahedral mesh of variable element size by sphere packing over an unbounded 3D domain. *Computer Methods in Applied Mechanics and Engineering*, 194 (48–49):5002–5018, 2005.

Pingchuan Ma, Tao Du, John Z. Zhang, Kui Wu, Andrew Spielberg, Robert K. Katzschmann, and Wojciech Matusik. DiffAqua: A differentiable computational design pipeline for soft underwater swimmers with shape interpolation. *ACM Transactions on Graphics*, 40(4):132:1–132:14, 2021.

Sehee Min, Jungdam Won, Seunghwan Lee, Jungnam Park, and Jehee Lee. SoftCon: Simulation and control of soft-bodied animals with biomimetic actuators. *ACM Transactions on Graphics*, 38(6):208:1–208:12, 2019.

Corey S. O'Hern, Leonardo E. Silbert, Andrea J. Liu, and Sidney R. Nagel. Jamming at zero temperature and zero applied stress: The epitome of disorder. *Physical Review E*, 68(1):011306, 2003.

Per-Olof Persson and Gilbert Strang. A simple mesh generator in MATLAB. *SIAM Review*, 46(2): 329–345, 2004.

Nan Qi, Yufeng Nie, and Weiwei Zhang. Acceleration strategies based on an improved bubble packing method. *Communications in Computational Physics*, 16(1):115–135, 2014. doi: 10.4208/cicp.080213.151113a.

Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In *Proceedings of SIGGRAPH*, pp. 151–160, 1986.

Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3D shape synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 6087–6101, 2021.

Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible isosurface extraction for gradient-based mesh optimization. *ACM Transactions on Graphics*, 42(4), 2023.

Jonathan Richard Shewchuk. Tetrahedral mesh generation by Delaunay refinement. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry (SoCG)*, pp. 86–95, 1998.

Kenji Shimada and David C. Gossard. Bubble mesh: Automated triangular meshing of non-manifold geometry by sphere packing. In *Proceedings of the Third ACM Symposium on Solid Modeling and Applications*, pp. 409–419, 1995.

Hang Si. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software*, 41(2):11:1–11:36, 2015. doi: 10.1145/2629697.

Karl Sims. Evolving virtual creatures. In *Proceedings of SIGGRAPH*, pp. 15–22, 1994.

Luke Strgar and Sam Kriegman. Accelerated co-design of robots through morphological pretraining. In *International Conference on Learning Representations (ICLR)*, 2026.

Luke Strgar, David Matthews, Tyler Hummer, and Sam Kriegman. Evolution and learning in differentiable robots. In *Robotics: Science and Systems (RSS)*, 2024.

D. Sulsky, Z. Chen, and H. L. Schreyer. A particle method for history-dependent materials. *Computer Methods in Applied Mechanics and Engineering*, 118(1–2):179–196, 1994. doi: 10.1016/0045-7825(94)90112-0.

Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH*, pp. 351–358, 1995.

Tsun-Hsuan Wang, Pingchuan Ma, Andrew Everett Spielberg, Zhou Xian, Hao Zhang, Joshua B. Tenenbaum, Daniela Rus, and Chuang Gan. SoftZoo: A soft robot co-design benchmark for locomotion in diverse environments. In *International Conference on Learning Representations (ICLR)*, 2023.

Soji Yamakawa and Kenji Shimada. High quality anisotropic tetrahedral mesh generation via ellipsoidal bubble packing. In *Proceedings of the 9th International Meshing Roundtable (IMR)*, pp. 263–273, 2000.

A SOFT-ROBOT DESIGN REPRESENTATIONS

Table A1: Body representations in soft-robot design. Free-form: ✓ any closed level-set surface; ∼ shape made of grid cells or particles; × primitives or a template. Diff.: simulator gradients.

Work	Representation	Surface	Free-form	Task	Compute	Diff.
Sims (1994)	primitives	blocky	×	diverse	CM-5	×
Cheney et al. (2013)	voxels (10^3)	blocky	∼	forward locomotion	VoxCAD / CPU	×
EvoGym (2021)	voxels (5×5)	blocky	∼	diverse (2D)	C++ / CPU	×
DiffAqua (2021)	voxel grid	blocky	×	forward swim	DiffPD / CPU	✓
SoftZoo (2023)	MPM particles	particle	∼	directional locomotion	Taichi / GPU	✓
Ours	conforming tet mesh	smooth	✓	swimming, walking	JAX / GPU	✓

Table A1 compares the body representation of our design pipeline with earlier soft-robot design systems. Primitive and voxel bodies have blocky surfaces, and particle bodies have no explicit surface. Each of these systems restricts the body to primitives, a template or a grid, and two of them provide simulator gradients. Our body is a conforming tetrahedral mesh with a smooth surface, built from a closed level-set surface, and the simulator provides gradients with respect to its nodes.

B STATIC MESHING ON NINE SHAPES

Figure A4 shows seven of the nine shapes of Section 4.1, and Table A2 lists the statistics of all nine. Our Vol CV lies in 0.127–0.207, against 0.293–0.308 for TetGen and 0.310–0.540 for fTetWild, and no mesher produces an inverted element. Our mean AMIPS energy is 3.32–3.47, the lowest or second lowest on every shape. Our pipeline takes 35.9–123.4 s per shape, TetGen under one second and fTetWild 6.7–1340 s. The I-beam gives our highest Vol CV and the dragon our longest run.

Table A2: Static mesh statistics at 64k interior spheres for the nine shapes of Figure 3. Nodes and Tets: our meshes; shading per shape; no inverted elements for any method.

			Vol CV ↓			AMIPS ↓			Wall (s)		
Shape	Nodes	Tets	Ours	TetGen	fTetWild	Ours	TetGen	fTetWild	Ours	TetGen	fTetWild
Sphere	74,242	415,041	0.127	0.297	0.441	3.33	3.81	3.41	36.8	0.4	16.2
Cube	76,636	421,981	0.130	0.298	0.310	3.32	3.81	3.31	35.9	0.5	7.4
Beam	87,379	455,294	0.155	0.293	0.324	3.37	3.79	3.42	41.2	0.6	55.6
I-beam	104,083	511,809	0.207	0.308	0.355	3.47	3.81	3.49	44.6	0.8	6.7
Spot	78,577	427,679	0.134	0.296	0.332	3.33	3.80	3.49	36.4	0.5	50.3
Bob	79,513	430,960	0.140	0.300	0.329	3.34	3.82	3.49	37.0	0.5	52.0
Bunny	77,726	425,192	0.134	0.296	0.540	3.33	3.81	3.33	36.9	0.5	77.2
Armadillo	82,870	437,233	0.149	0.298	0.342	3.35	3.81	3.41	41.0	0.6	1340.0
Dragon	89,507	455,341	0.167	0.300	0.339	3.39	3.81	3.42	123.4	0.7	887.2

C SIMULATOR SPEED

Table A3: Simulator speed on one RTX 5090 for a 6 s rollout (24,000 steps); median of five calls after a warm-up, compilation excluded; memory: peak GPU memory, forward / with gradient.

Mesh	Nodes	Tets	Forward (s)	Steps/s	With gradient (s)	Memory (MiB)
Capsule, 1k	1,680	7,954	0.40	60,600	3.56	654 / 1,678
Capsule, 8k	10,768	56,438	0.79	30,200	6.28	656 / 8,852
Bunny, 64k	77,726	425,192	2.27	10,600	–	760 / –

Table A3 times the simulator of Section 3.4 in the swimming setting of Section 4.3 on one NVIDIA RTX 5090, with JAX 0.9 in single precision. A 6 s rollout of the 1k capsule runs in 0.40 s, 15

times faster than real time, and its gradient with respect to all rest positions takes 3.56 s, 9.0 times the forward cost. The 8k capsule takes 0.79 s forward and 6.28 s with its gradient, 7.9 times the forward cost. The forward rollout of the 64k bunny takes 2.27 s; its gradient was not run, because the compiled program was estimated to need more than 82% of the card's 32 GB. The benchmark uses checkpoint blocks of 500 steps, and compilation takes 2.8–13.4 s.

Gradient checks. At a fixed mesh, the adjoint gradient of a rollout with respect to element stiffness agrees with central finite differences to 5.5×10^{-9} in relative terms without ground contact. With contact, the two depart as the difference step shrinks, with relative errors of 0.19, 0.55 and 0.97 at $\epsilon = 10^{-2}, 10^{-3}, 10^{-4}$, so the walking objective is non-smooth. Repeated gradient evaluations on one GPU differ by 2.3×10^{-3} in relative terms, because unordered scatter-adds run as atomics.

D ROBOT DESIGNS

Gait. A plane splits the muscle edges into two groups, which contract in antiphase: the rest length of a muscle edge is scaled by $a_e(t) = 1 \pm A \sin\big(2\pi(ft - \xi_e)\big)$ with $f = 2$ Hz, where the phase ξ_e grows by one period from head to tail; $A = 0.18$ for swimming and $A = 0.12$ for walking. Swimming uses the drag model of Section 3.4 without gravity, and walking uses frictional ground.

Setup. All quantities are in simulation units. The simulator steps at 0.25 ms, 24,000 steps per 6 s rollout, with node mass 0.0125, edge stiffness 4,000, edge damping 3 and Neo-Hookean modulus 4,000; swimming uses normal and tangential drag coefficients of 480 and 80. The design objective adds penalties on lateral drift and body orientation to the fitness. Each gradient averages eight rollouts of perturbed shapes, four antithetic pairs with a perturbation RMS of 0.03. The FFD lattice spacing is 8 sphere spacings for swimming and 12 for walking at 1k, and 16 and 24 at 8k, where the spheres lie half as far apart; the lattice has 5^3 control points for swimming and 4^3 for walking. Four of the 90 runs ended early on a CDT failure after their best fitness had plateaued, and each is scored by the best fitness it reached.

Controllers. Following the clock inputs of ELDiR (Strgar et al., 2024), a shared multilayer perceptron maps eight clock signals and the rest position of each of $2S$ muscle blocks to that block's activation, for $S \in \{2, 4, 8, 16\}$. It reads no simulation state and, as in ChainQueen (Hu et al., 2019), is trained by gradient descent through the simulator, with one training run and the same budget for each method and size. The least-squares initialization fits the travelling wave with normalized error 0.725 at S=2 and 0.114 at S=16. The six test points of Figure 7 are the design stage, the fixed gait with tuned amplitude and period, and the four controller sizes; with the tuned gait, ours reaches 1.64× the fitness of ours-surface.

Figure A1 shows the final design of the best seed in every swimming condition, grown from the same capsule, 1.91 long with a 0.72×0.72 cross-section; Section 4.3 compares their shapes.

Figure A2 plays back the best 1k up–down swimmers of remesh-fTetWild and ours under the design-stage wave, with amplitude 0.18, and under the learned controller with S=16, which drives 32 muscle blocks at its training amplitude of 0.24; Section 4.3 gives their distances. The camera follows each body at a fixed offset, and the floor grid shows the distance travelled.

E VOXEL AND SMOOTH SURFACES

Table A4: Voxel re-evaluation of eight of our swimming designs from an earlier design sweep. Fitness: ours, the stored training fitness; voxel, median over three start phases; mean±sd over the n designs; Δ: change from ours; grid nodes relative to ours in brackets.

			Ours (smooth)		Voxel, matched triangles			Voxel, 128^3 grid		
Environment	Scale	n	Fitness	Nodes	Fitness	Δ	Nodes	Fitness	Δ	Nodes
swim, up–down	1k	3	1.856 ± 1.249	1,965	0.830 ± 0.617	−55%	1,485	1.257 ± 0.832	−32%	28,420 (14.5×)
swim, lateral	1k	3	2.402 ± 0.054	2,026	1.261 ± 0.040	−48%	1,536	1.586 ± 0.041	−34%	29,913 (14.8×)
swim, up–down	8k	1	2.436	11,712	1.535	−37%	9,198	1.286	−47%	30,818 (2.6×)
swim, lateral	8k	1	2.618	12,008	1.327	−49%	8,842	0.968	−63%	32,548 (2.7×)

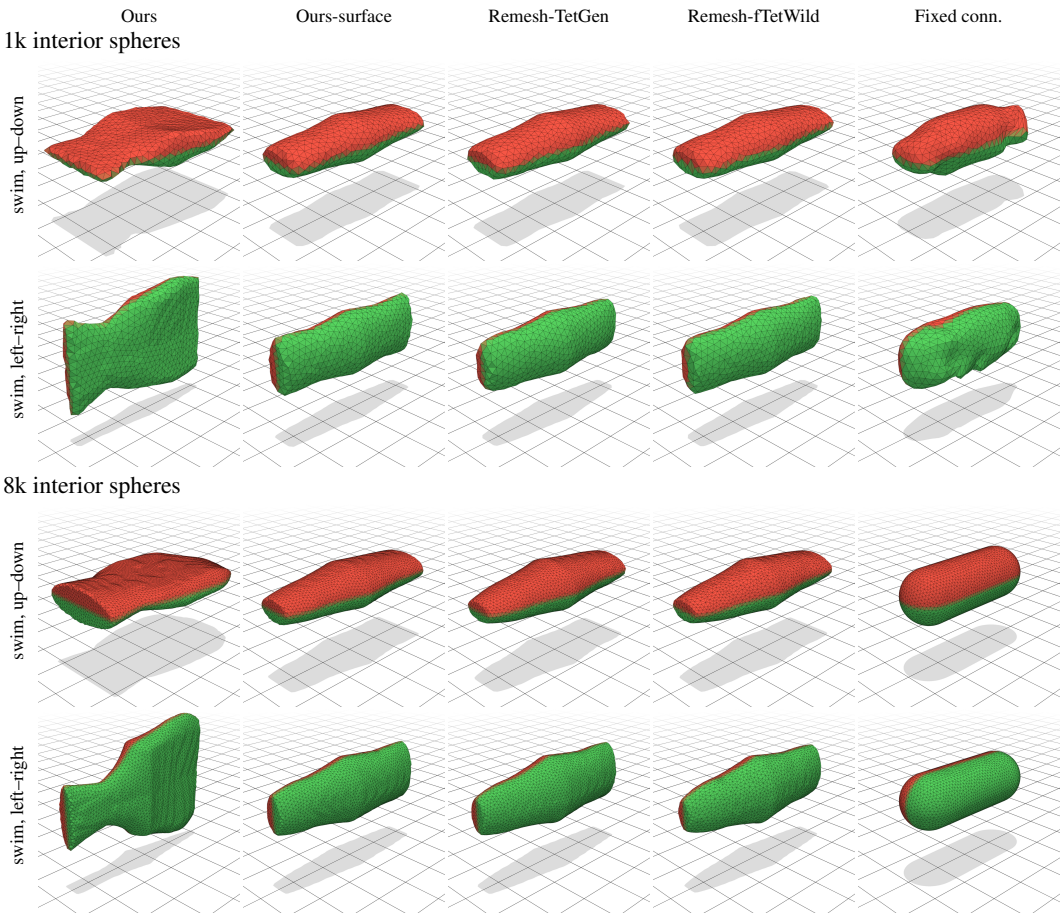


Figure A1: Final design of the best seed in every swimming condition of Table 3, at rest, coloured by muscle group; columns: methods; rows: up–down and left–right swimming; top block 1k, bottom block 8k interior spheres; same scale in all panels.

Table A4 gives the voxel comparison of Section 4.3 in full. It uses eight of our designs from an earlier sweep, distinct from the seeds of Table 3: three per swimming condition at 1k and one at 8k. The smooth reference is the stored training fitness of each design, and each voxel body is scored by the median fitness over three start phases. Each design keeps its shape, muscle rule and fitness measure, and its tetrahedral body is replaced by a voxel body at one of two resolutions. The first has as many surface triangles as our mesh and 0.74–0.79$\times$ our node count; the second is the 128^3 level-set grid of the design loop. The voxel bodies lose 32–63% of the smooth designs' fitness.

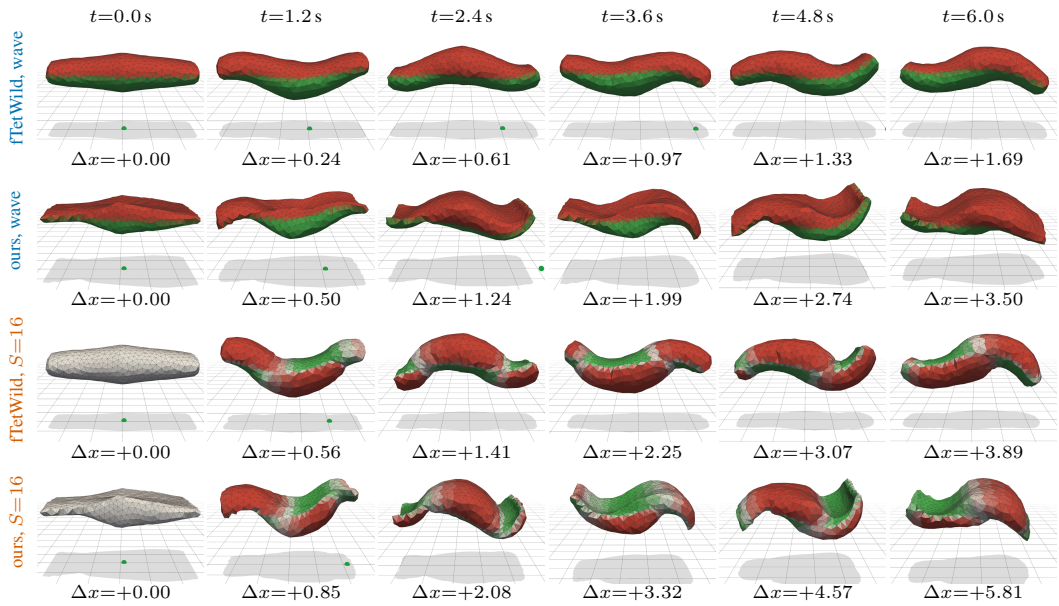


Figure A2: Best 1k up–down swimmers of remesh-fTetWild and ours, rendered as in Figure 6. Rows 1–2: design-stage wave, coloured by muscle group; rows 3–4: learned controller with S=16, coloured by muscle activation; Δx: forward displacement at time t; green dot: start position.

F Shape design with linear finite elements

The meshes also support gradient-based design with linear finite elements. We minimize the compliance $C = \mathbf{f}^\top \mathbf{u}$ of a beam under a static 100 N load at fixed volume, with aluminium properties ($E = 68.9$ GPa, $\nu = 0.33$). The design variable is a section scale $s(z) \in [0.55, 1.75]$ along the beam axis, set by 12 segments through a sigmoid and blended by Gaussian weights along z; the cross-section is rescaled after each step so that the volume stays fixed. The cantilevers are clamped at the base and carry the load on the tip nodes; the beam clamped at both ends has both end faces pinned transversally and carries the load at mid-span. The gradient comes from the adjoint of the static solve, $\partial C/\partial p = -\mathbf{u}^\top(\partial K/\partial p)\,\mathbf{u}$, and Adam takes 60 steps. Our beam meshes use 8k interior spheres (11.5k–16.7k nodes), and the TetGen and fTetWild meshes of the short beams have 10.7k–12.9k nodes. Compliance falls by 30.4–34.1% (Table A5); on the short cantilever it drops from 1.86 to 1.28 mJ, with 90% of the drop reached by step 14. All four cantilevers taper from a section scale of 1.48–1.54 at the root to 0.63–0.64 at the tip, and their profiles have an RMS deviation of 0.067–0.069 from the uniform-strength profile $s \propto (L - z)^{1/3}$, at both aspect ratios. On the beam clamped at both ends the RMS deviation from $|M(z)|^{1/3}$ is 0.131, because the lower bound $s \geq 0.55$ holds the profile above zero where the bending moment M vanishes (Figure A3). On the two short beams, TetGen and fTetWild meshes of the same shape reach the same compliance reduction within 0.2 percentage points, and their profiles differ from ours by an RMS of 0.001–0.010.

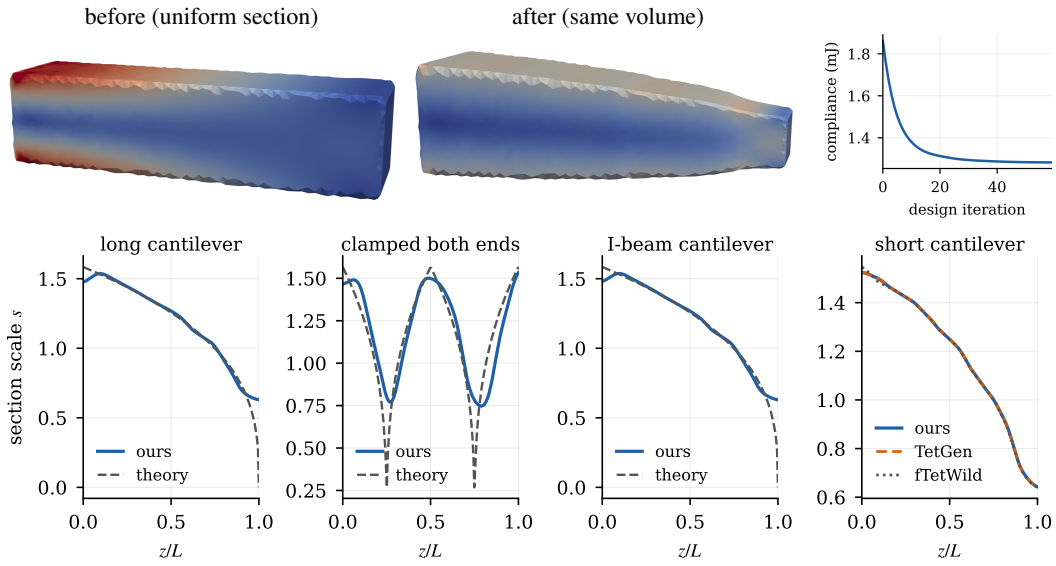


Figure A3: Top: compliance design of a short cantilever (1:1:4, tip load), uniform section and optimized shape at equal volume, coloured by von Mises stress, 0 5.3 MPa; right: compliance per design step. Bottom: optimized section scale along the beam against the analytical profile; short cantilever: the same design on TetGen and fTetWild meshes.

Table A5: Compliance before (C_0) and after (C_1) 60 design steps at fixed volume. Mesh: the volume mesher, at 10.7k–16.7k nodes.

Structure	Mesh	C_0 (mJ)	C_1 (mJ)	ΔC
Cantilever, 1:1:20	ours	210.86	139.10	−34.0%
Cantilever, 1:1:4	ours	1.86	1.28	−31.1%
	TetGen	1.85	1.27	−31.1%
	fTetWild	1.83	1.26	−31.3%
Clamped both ends, 1:1:20	ours	3.68	2.56	−30.4%
I-beam cantilever, 1:1:20	ours	330.39	217.64	−34.1%
I-beam cantilever, 1:1:4	ours	3.11	2.14	−31.2%
	TetGen	3.08	2.12	−31.2%
	fTetWild	2.96	2.04	−31.0%

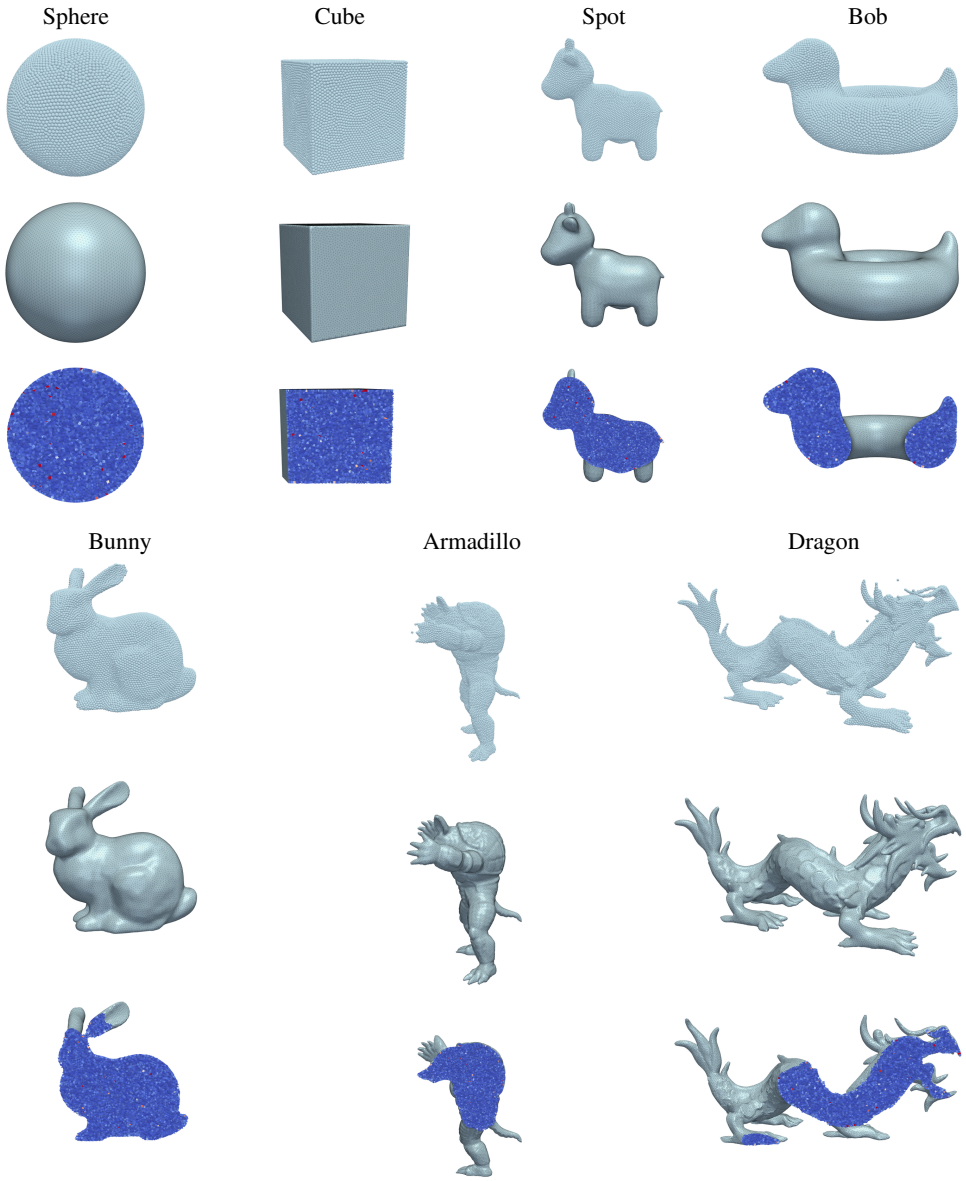


Figure A4: Seven shapes at 64k interior spheres. Rows per block: packed spheres; surface mesh; cut view coloured by AMIPS energy: regular distorted.