RigidFormer: Learning Rigid Dynamics using Transformers

Zhiyang Dou[1] **Minghao Guo**[1] **Haixu Wu**[1] **Doug Roble**[2]
Tuur Stuyck[2] **Wojciech Matusik**[1]

[1] MIT [2] Meta

Abstract

Learning-based simulation of multi-object rigid-body dynamics remains difficult because contact is discontinuous and errors compound over long horizons. Most existing methods remain tied to mesh connectivity and vertex-level message passing, which limits their applicability to mesh-free inputs such as point clouds and leads to high computational cost. Efficiently modeling high-fidelity rigid-body dynamics from mesh-free representations therefore remains challenging. We introduce **RigidFormer**, an object-centric Transformer-based model that learns mesh-free rigid-body dynamics with controllable integration step sizes. RigidFormer reasons at the *object level* and advances each object through compact anchors; *Anchor-Vertex Pooling* enriches these anchors with local vertex features, retaining contact-relevant geometry without dense vertex-level interaction. We propose *Anchor-based RoPE* to make attention aware of the objects' spatial layout while respecting the unordered nature of objects and anchors: object-token processing is permutation-equivariant, and the mean-pooled anchor descriptor is invariant to anchor reindexing. RigidFormer further enforces *rigidity* by projecting updates onto the rigid-body manifold using differentiable Kabsch alignment. On standard benchmarks, RigidFormer outperforms or matches mesh-based baselines using point inputs, runs faster, generalizes to unseen point resolutions and across datasets, and scales to 200+ objects; we also show a preliminary extension to command-conditioned articulated bodies by treating body parts as interacting object-level components. Project page: frank-zy-dou.github.io/projects/RigidFormer

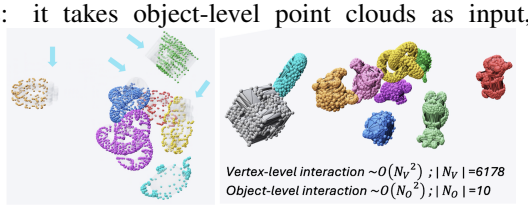
1 Introduction

Rigid-body dynamics arises throughout robotics, graphics, and embodied AI. With accurate meshes, reliable physical parameters, and well-tuned contact models, classical physics engines [9, 12, 24, 25, 35] can produce faithful trajectories. In practice, however, these prerequisites are often missing: objects may only be available as imperfect or incomplete geometry (e.g., polygon soups or point clouds), with contact properties that are difficult to calibrate. This motivates *mesh-free* modeling. A common choice is a point-based representation, which is easy to acquire, topology-free, and resolution-flexible, making it a natural interface between perception and dynamic scene modeling [7, 22, 30, 31, 38, 47, 48]. It also integrates naturally with modern generative pipelines [18, 42, 46, 49]. However, despite the appeal of point-based inputs, many state-of-the-art learned simulators remain mesh-dependent [1, 29, 33, 39, 43], requiring explicit edge and face connectivity that is not available for point inputs. Moreover, since they typically operate at the vertex, edge, or facet level, their computational costs grow rapidly with resolution, significantly limiting inference efficiency.

We present RigidFormer, an object-centric Transformer-based model that learns mesh-free multi-object rigid-body dynamics. Our design is guided by three observations. First, a rigid body responds to an impulse as a coherent whole: interaction effects do not need

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

to "diffuse" across surface vertices edge by edge, as in vertex-centric simulators based on local message passing [1, 29, 33], which can introduce substantial computational overhead and slow down inference. RigidFormer therefore adopts an object-centric representation that reasons over objects rather than vertices: it takes object-level point clouds as input, even partial ones (see the inset and Sec. 4.4), and encodes each object into a compact token without requiring connectivity. Interactions are then modeled primarily among *object tokens*, matching the rigid-body assumption that each object moves as a coherent whole. This shift greatly improves efficiency, e.g., 83.6 FPS versus 3.0 FPS, while maintaining simulation quality. We implement this design with Transformers [36], whose attention mechanism flexibly captures multi-object interactions without relying on hand-designed graphs.



(a) Partial Point Clouds (b) Reduced Complexity

Second, we exploit the low-dimensional structure of rigid-body motion during state advancement. Although an object may contain thousands of points, rigid motion lives in a low-dimensional space (6-DoF per object). Therefore, we solve for each object's state update using a small number of *Anchors*, which enables efficient and geometry-aware dynamics updates. Anchor/keypoint and $SE(3)$ representations have precedent in pose tracking and manipulation models [5, 37]; here, anchors serve as learned simulation states for long-horizon contact dynamics rather than tracked category keypoints or directly regressed part transforms. Given the central role of positional embeddings in Transformer generalization [16, 34, 45], we propose an *Anchor-based Rotary Positional Embedding (ARoPE)* (Sec. 3.3) to make attention aware of the objects' spatial layout. Its symmetry properties are deliberately scoped: the object Transformer remains equivariant to object-token permutations because no sequence-index positional embeddings are used, while the mean-pooled ARoPE descriptor is invariant to anchor reindexing within each object. A distance-kernel *Anchor-Vertex Pooling* module further supplies local contact geometry with vertex-order-invariant aggregation. Rather than directly regressing rotations and translations, which can be error-prone [51], we predict anchor motions and obtain the rigid transform via *differentiable rigid projection*, which projects updates onto the rigid-body manifold while preserving gradient flow, improving long-horizon stability. Finally, inspired by time-conditioned neural simulation [8, 44, 50], RigidFormer conditions on the temporal discretization, enabling a single model to operate across step sizes (Sec. 4.1). Larger Δt improves long-horizon accuracy by reducing autoregressive error accumulation, while smaller Δt captures finer temporal detail when needed.

Table 1: **Method comparison.** Mesh-free: no triangle connectivity required. Variable step: handles multiple Δt. Preproc.-free: no offline geometry computation. Warmup frames: number of input frames required for rollout.

Method	Mesh-Free	Var. Δt	Preproc.-Free	#Warmup Frames ↓	Runtime (FPS) ↑
MGN	✗	✗	✓	**2**	5.7
FIGNet	✗	✗	✓	3	3.0
SDF-Sim	✗	✗	✗[†]	3	–
HopNet	✗	✗	✗[‡]	3	0.2
Ours	✓	✓	✓	**2**	**83.6**

[†] SDF pre-learning: ∼5 hours for MOVi-B.
[‡] Simplicial-complex construction: ∼15 days on MOVi-B.

As a result, RigidFormer offers an efficient, stable, and scalable framework for rigid-body dynamics modeling; a comparison with previous methods is in Tab. 1. On MOVi [15], it matches or surpasses prior mesh-based methods using only point positions, without requiring mesh connectivity. It leverages known material parameters when available, generalizes to unseen point resolutions and across datasets, supports step-size control, and scales beyond 200 objects while maintaining both accuracy and high inference speed. RigidFormer also handles partial point-cloud inputs (see Fig. A5). We further show a preliminary application of the same object-anchor design to controllable articulated bodies by treating body parts as interacting object-level components. We summarize our contributions below:

- We introduce an efficient and scalable mesh-free Transformer-based neural simulator named RigidFormer for multi-object rigid-body dynamics from point representations, supporting simulation across different time-step sizes.
- We propose an object-level formulation with Anchor-based RoPE for position-aware attention with explicit object-token permutation equivariance and anchor-order invariance, along with vertex-order-invariant Anchor-Vertex Pooling and a low-dimensional anchor state advance that reduces complexity; rigidity and long-horizon stability are enforced via projection onto the rigid-body manifold during the simulation.
- We validate RigidFormer in diverse experiments, showing fast inference, generalization, scalability, partial point-cloud input, and a preliminary application to command-conditioned articulated bodies.

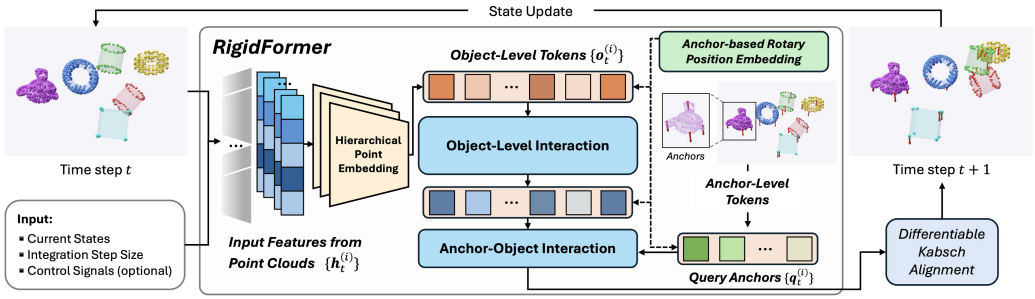


Figure 1: **RigidFormer Pipeline.** Inputs include two recent point-level states, the time-step size, and optional control signals. At time t, we encode each object point cloud into an object token. *Object-Level Interaction* models direct object–object effects, while *Anchor-Object Interaction* lets anchors attend to multi-scale object features and cross-object context. Each object's state is advanced efficiently using a compact set of anchors. We integrate anchor dynamics using Verlet integration followed by differentiable Kabsch alignment to enforce rigidity and obtain the state at $t{+}1$. Anchor-based RoPE is used to improve generalization across object counts.

2 Related Work

Classical numerical rigid-body simulators [2, 9, 25, 35] resolve contact by solving constrained optimization or complementarity problems. Differentiable simulators (e.g., DiffTaichi [17], Warp [24], and Brax [12]) enable gradient-based learning and inverse problems, but they rely on explicit physics engines and typically assume mesh-based geometry rather than mesh-free point inputs.

Early learning-based dynamics models often targeted relatively simple systems with explicit, low-dimensional state representations, typically in 2D. Interaction Networks [3] and Neural Physics Engine [6] established object- and relation-centric inductive biases and motivated graph-based simulators for dynamics modeling. In rigid-body systems, state-of-the-art neural simulators typically rely on mesh-based inputs in order to faithfully capture the dynamics [1, 29, 33, 39]. MeshGraphNets [29] extend message passing to mesh discretizations and achieve strong performance for mesh-based simulation. FIGNet [1] improves collision modeling by constructing interactions over mesh faces rather than nodes. HopNet [39] incorporates higher-order topology and physics-informed message passing for rigid interactions; however, obtaining the required topological structures can be expensive. HCMT [43] uses hierarchical mesh structures and Transformer-style long-range modeling for collision-induced dependencies in flexible-body collision dynamics in the 2D domain. All aforementioned methods require mesh connectivity and incur substantial vertex-level interaction cost as resolution grows. SDF-Sim [33] represents shapes with learned signed distance functions, reducing collision-handling bottlenecks but requiring additional shape learning. Compared to these works, RigidFormer is mesh-free: it models rigid-body dynamics using point inputs, shifts interaction reasoning to the object level, and uses an anchor-based advance to avoid dense vertex interactions at inference time; see Tab. 1. Rigid-motion and keypoint-based representations have also appeared in related robotics settings. SE3-Nets [5] predict rigid $SE(3)$ transforms for object parts from point clouds and action inputs, demonstrating the value of rigid-motion inductive bias for manipulation. 6-PACK [37] learns anchor-based 3D keypoints for category-level 6D pose tracking. These works are complementary to ours: their keypoints are used for pose estimation in manipulation or tracking, whereas ours are for more dynamic simulation states that are advanced by learned dynamics, coupled with ARoPE and differentiable rigid projection for long-horizon multi-object rollout.

Point-based representations for dynamics have been explored recently. Kim and Li [20] propose a hierarchical point-cloud representation with continuous point convolutions to improve contact accuracy. Whitney et al. [40, 41] learn point-based dynamics by disentangling visual observations from physical states, but accuracy degrades in contact-rich regimes due to the coupling. In contrast, RigidFormer adopts an object-level Transformer that effectively models inter-object interactions for points, leading to higher-quality dynamics prediction.

3 Methodology

We consider a system of M rigid objects, where M can vary across scenes during both training and inference. Object i is represented by a point set $\mathbf{x}_t^{(i)} \in \mathbb{R}^{N_v^{(i)} \times d}$ at time t, where $N_v^{(i)}$ is the number of points and d denotes the per-point feature dimension, and the full state is $\mathbf{x}_t = \{\mathbf{x}_t^{(i)}\}_{i=1}^{M}$. We learn

a model f_θ that updates the next state from two consecutive observations: $\mathbf{x}_{t+1} = f_\theta(\mathbf{x}_{t-1}, \mathbf{x}_t, \Delta t)$. An overview of our pipeline is in Fig. 1.

3.1 Object-Centric Interaction Modeling

We concatenate, for each vertex of object i: (1) the nearest-neighbor displacement vector $\mathbf{d}_t^{(i)} \in \mathbb{R}^{N_v^{(i)} \times 3}$ from the vertex to the nearest point on another object or the ground plane, (2) the velocity $\mathbf{v}_t^{(i)} = \mathbf{x}_t^{(i)} - \mathbf{x}_{t-1}^{(i)} \in \mathbb{R}^{N_v^{(i)} \times 3}$, (3) the reference-offset $\mathbf{r}_t^{(i)} = \mathbf{x}_t^{(i)} - \mathbf{x}_{\text{ref}}^{(i)} \in \mathbb{R}^{N_v^{(i)} \times 3}$ where the reference is the first frame of the trajectory, and (4) physics parameters $\boldsymbol{\phi}^{(i)} = [m, \mu, \epsilon] \in \mathbb{R}^3$ (mass, friction, restitution). These yield the input feature $\mathbf{h}_t^{(i)} \in \mathbb{R}^{N_v^{(i)} \times 12}$. Inspired by PointNet [30], we build an encoder Enc_θ with hierarchical feature extraction to aggregate per-vertex features into a fixed-dimensional object embedding: $\mathbf{o}_t^{(i)} = \mathrm{Enc}_\theta(\mathbf{h}_t^{(i)}) \in \mathbb{R}^D$. After computing per-vertex features, we extract multi-scale geometry at the global level and three subsampled levels; these features are then concatenated and fused into one object-level embedding. This design captures both fine-grained local geometry and coarse global structure while remaining robust to variable vertex counts (see validation in Sec. 4.1). The encoder is *shared* across all objects, promoting generalization to various geometries. Using one token per object drastically shortens the Transformer sequence compared with vertex-level modeling, substantially improving efficiency; see Appendix F.1.

Given object embeddings $\mathbf{O}_t = [\mathbf{o}_t^{(1)}, \ldots, \mathbf{o}_t^{(M)}] \in \mathbb{R}^{M \times D}$, our decoder stacks L Transformer blocks initialized from $\mathbf{O}_t$ and N_r=16 learned register tokens. For clarity, we describe the object-token update and omit the registers from notation. At layer ℓ, the decoder applies residual self-attention, step-size FiLM conditioning [28], and a residual feed-forward update: $\tilde{\mathbf{Z}}_t^{(\ell)} = \mathrm{SelfAttn}(\mathbf{Z}_t^{(\ell-1)}) + \mathbf{Z}_t^{(\ell-1)}$, $\hat{\mathbf{Z}}_t^{(\ell)} = \boldsymbol{\gamma}_\ell(\mathbf{c}) \odot \tilde{\mathbf{Z}}_t^{(\ell)} + \boldsymbol{\beta}_\ell(\mathbf{c})$, and $\mathbf{Z}_t^{(\ell)} = \mathrm{FFN}(\hat{\mathbf{Z}}_t^{(\ell)}) + \hat{\mathbf{Z}}_t^{(\ell)}$, for $\ell = 1, \ldots, L$. The FiLM code $\mathbf{c} = (s, s^2) \in \mathbb{R}^2$ encodes the integration step size, where $s \propto \Delta t$ captures first-order time scaling and s^2 mirrors the Δt^2 factor in Verlet integration. The layer-specific MLPs $\boldsymbol{\gamma}_\ell$ and $\boldsymbol{\beta}_\ell$ produce channel-wise scale and shift parameters, allowing the same decoder to adapt its features across different temporal discretizations. Motivated by gated attention in language modeling [32], we modulate each attention update with a query-conditioned sigmoid gate: $\mathbf{y} = \sigma(\mathbf{G}(\mathbf{Q})) \odot \mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$, where $\mathbf{G}(\mathbf{Q})$ has the same per-head channel shape as the attention output. In our setting, the gate acts as a learned attenuator for noisy or weakly relevant interaction reads, which stabilizes autoregressive dynamics rollouts and improves long-horizon accuracy (Sec. 4.2). The output of this stage is a set of updated object-level tokens $\{\mathbf{Z}_t^{(L)}\}$ that summarize scene context and object interactions. We next use anchors as queries into these object tokens, combine the retrieved context with local vertex features, and advance the system through anchor dynamics.

3.2 Anchor-based State Advance

Rigid-body motion is low-dimensional (6-DoF) even when an object contains thousands of points. Therefore, directly updating all vertices is expensive and redundant: dense attention over MN_v points costs $O((MN_v)^2)$, and per-vertex regression destabilizes prediction. Meanwhile, regressing rotations and translations directly [51] can be error-prone and unstable due to discontinuities in common $SE(3)$ parameterizations. In RigidFormer, we instead select a small set of N_a=4 *anchors* per object using FPS [14, 31], reducing the interaction cost to $O((MN_a)^2)$ with $N_a \ll N_v$ across time steps. We form anchor queries by extracting features at anchor locations and projecting them with an MLP to obtain $\mathbf{Q}_t \in \mathbb{R}^{(MN_a) \times D}$. Each anchor query attends to the updated object tokens $\{\mathbf{Z}_t^{(L)}\}$ via cross-attention, retrieving object-level interaction context, and the network then predicts a per-anchor acceleration $\mathbf{a}_t^{(i,k)}$.

Anchor queries summarize a rigid object's dynamics, but accurate acceleration prediction during contact depends strongly on which vertices lie close to the contact site. To inject this fine-grained, contact-local geometry into each anchor without paying the cost of full per-vertex attention, we attach an Anchor-Vertex Pooling (AVP) module that aggregates per-vertex encoder features around each anchor with a learnable isotropic distance kernel:

$$\mathbf{u}_t^{(i,k)} = \frac{\sum_{v=1}^{N_v^{(i)}} w_t^{(i,k,v)}\, \mathbf{f}_t^{(i,v)}}{\sum_{v=1}^{N_v^{(i)}} w_t^{(i,k,v)}}, \quad w_t^{(i,k,v)} = \exp\left(-\frac{\|\mathbf{x}_t^{(i,v)} - \mathbf{q}_t^{(i,k)}\|}{\sigma}\right), \tag{1}$$

where $\mathbf{f}_t^{(i,v)}$ is the encoder feature at vertex v, $\mathbf{q}_t^{(i,k)}$ is the position of anchor k, and the kernel bandwidth σ is trained jointly with the rest of the network; padded vertices are masked in the batched implementation. Because the weights depend only on point-anchor distances and the aggregation is a normalized sum, AVP is invariant to vertex ordering, and its attention weights are unchanged by a common rigid transform of the point and anchor coordinates. The resulting feature $\mathbf{u}_t^{(i,k)} \in \mathbb{R}^{256}$ is concatenated to the anchor query before predicting acceleration, enriching it with collision-aware local context. We advance anchors with Verlet integration to obtain candidate anchor positions from predicted accelerations: $\hat{\mathbf{q}}_{t+1}^{(i,k)} = \mathbf{a}_t^{(i,k)} \Delta t^2 + 2\mathbf{q}_t^{(i,k)} - \mathbf{q}_{t-1}^{(i,k)}$.

Scatter to All Vertices via Differentiable Rigid Projection. We recover the rigid transform $(\mathbf{R}^{(i)}, \mathbf{t}^{(i)}) \in SE(3)$ by aligning reference anchors $\mathbf{q}_{\text{ref}}^{(i,k)}$ to the candidate anchors $\hat{\mathbf{q}}_{t+1}^{(i,k)}$ using Kabsch alignment [19]:

$$\begin{aligned} \mathbf{H} &= \sum_k (\mathbf{q}_{\text{ref}}^{(i,k)} - \bar{\mathbf{q}}_{\text{ref}}^{(i)})(\hat{\mathbf{q}}_{t+1}^{(i,k)} - \bar{\hat{\mathbf{q}}}^{(i)})^\top, \quad \mathbf{U}, \mathbf{\Sigma}, \mathbf{V}^\top = \text{SVD}(\mathbf{H}), \\ \mathbf{R}^{(i)} &= \mathbf{V}\,\text{diag}(1, 1, \det(\mathbf{V}\mathbf{U}^\top))\mathbf{U}^\top, \quad \mathbf{t}^{(i)} = \bar{\hat{\mathbf{q}}}^{(i)} - \mathbf{R}^{(i)} \bar{\mathbf{q}}_{\text{ref}}^{(i)}, \end{aligned} \tag{2}$$

where k indexes anchors, and $\bar{\hat{\mathbf{q}}}^{(i)}$ and $\bar{\mathbf{q}}_{\text{ref}}^{(i)}$ denote centroids of the predicted and reference anchor sets. Then, we update the full-resolution point set by broadcasting the transform to all vertices: $\mathbf{x}_{t+1}^{(i,v)} = \mathbf{R}^{(i)} \mathbf{x}_{\text{ref}}^{(i,v)} + \mathbf{t}^{(i)}, \quad \forall v \in \{1, \ldots, N_v\}$. This projection enforces rigidity by construction and improves long-horizon rollout stability. Since gradients through SVD can be unstable near degenerate singular values, we implement rigid registration with RoMa [4] for robust differentiability.

3.3 Anchor-based Rotary Positional Embedding

Effective positional embeddings are essential for the Transformer-based RigidFormer to generalize rigid-body dynamics, where interactions vary with object count and 3D geometry. In RigidFormer, object tokens have no inherent ordering—permuting objects should not change the predicted dynamics. Moreover, contact outcomes depend on relative 3D geometry rather than absolute indices. Encoding all vertices in attention is largely redundant, hurts generalization, and adds prohibitive computational overhead.

We propose *Anchor-based Rotary Positional Embedding (ARoPE)*, which encodes the spatial layout of objects from a sparse set of anchor positions, making the attention position-aware while remaining efficient and generalizable across different numbers of objects with various shapes. Concretely, for object i with anchors $\{\mathbf{x}_k^{(i)}\}_{k=1}^{N_a} \subset \mathbb{R}^3$ (N_a=4), we apply a shared 3D rotary anchor map $\psi_\omega(\cdot)$ to each anchor, mapping each coordinate through d_c=32 rotary phase channels to obtain a per-anchor 96-D phase descriptor. We then aggregate the per-anchor descriptors by mean-pooling:

$$\text{ARoPE}(\{\mathbf{x}_k^{(i)}\}) = \frac{1}{N_a} \sum_{k=1}^{N_a} \underbrace{\bigoplus_{j=1}^{3} \bigoplus_{l=1}^{d_c/2} \left[\omega_l x_{k,j}^{(i)},\, \omega_l x_{k,j}^{(i)}\right]}_{\psi_\omega(\mathbf{x}_k^{(i)})}, \tag{3}$$

where ω_l are log-spaced frequencies, $\oplus$ denotes concatenation, and the repeated terms form the even–odd channel pairs used by RoPE. The resulting descriptor provides the RoPE angles used in attention. For an attention head with query/key channels split into a rotary part and a pass-through part, $\mathbf{q} = [\mathbf{q}_r; \mathbf{q}_p]$ and $\mathbf{k} = [\mathbf{k}_r; \mathbf{k}_p]$, ARoPE applies

$$\tilde{\mathbf{q}} = [\mathbf{q}_r \odot \cos \mathbf{a}_q + \text{rot}(\mathbf{q}_r) \odot \sin \mathbf{a}_q; \mathbf{q}_p], \quad \tilde{\mathbf{k}} = [\mathbf{k}_r \odot \cos \mathbf{a}_k + \text{rot}(\mathbf{k}_r) \odot \sin \mathbf{a}_k; \mathbf{k}_p], \tag{4}$$

where $\mathbf{a}_q$ and $\mathbf{a}_k$ are the ARoPE descriptors for the query and key tokens, and $\text{rot}(\cdot)$ swaps each even–odd channel pair with a sign flip as in standard RoPE [34]. Mean-pooling these per-anchor rotary features—rather than concatenating raw anchor coordinates as in a naive multi-point variant—matches the symmetry that anchor identities are arbitrary. Since the rotary angles are linear in the coordinates, the mean-pooled descriptor equals the RoPE angles of the anchor centroid: ARoPE encodes where each object is, while its shape reaches attention through the object tokens and the AVP features. ARoPE is invariant to anchor reindexing: for any anchor permutation π, $\text{ARoPE}(\{\mathbf{x}_{\pi(k)}^{(i)}\}_{k=1}^{N_a}) = \frac{1}{N_a} \sum_{k=1}^{N_a} \psi_\omega(\mathbf{x}_{\pi(k)}^{(i)}) = \text{ARoPE}(\{\mathbf{x}_k^{(i)}\}_{k=1}^{N_a})$, because the sum is unchanged by reordering. Applying

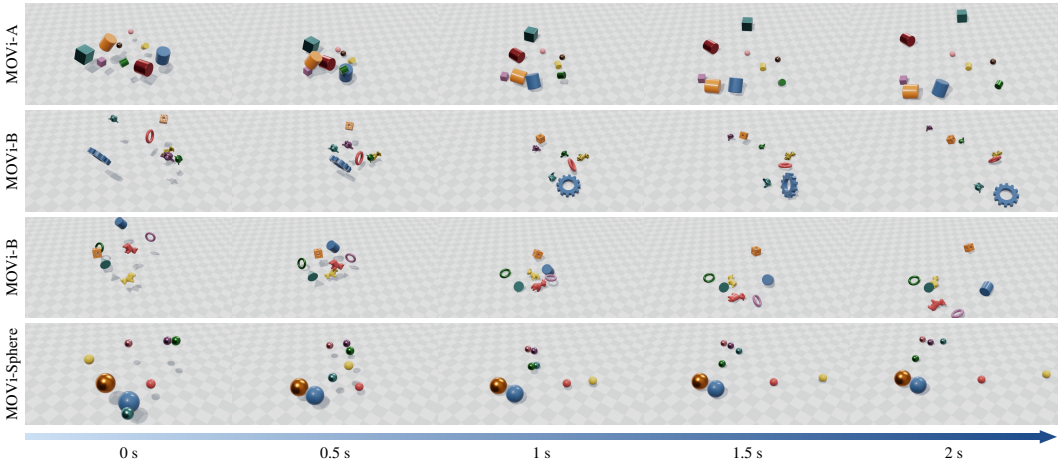


Figure 2: **Qualitative results from RigidFormer** on test scenes of MOVi-A, MOVi-B and MOVi-Sphere, from the initial state (0 s) to 2 s at step size 1. Meshes are shown only for visualization; our model operates on point inputs. Additional visualization results on the MOVi datasets are provided in Appendix A.

ARoPE to object tokens yields improved performance and generalization across varying object counts (Sec. 4.2). Detailed proofs and discussion are given in Appendix B.

3.4 Training Objectives

Our objective combines position and acceleration L1 losses: $\mathcal{L} = \lambda_{\text{pos}}(\mathcal{L}_{\text{pos}}^{\text{raw}} + \mathcal{L}_{\text{pos}}^{\text{rigid}}) + \lambda_{\text{acc}}(\mathcal{L}_{\text{acc}}^{\text{raw}} + \mathcal{L}_{\text{acc}}^{\text{rigid}})$, where "raw" and "rigid" denote losses computed before and after Kabsch alignment, respectively, with λ_{pos}=10 and λ_{acc}=1, applied to a Smooth L1 reduction [13]. All four terms are supervised at the selected anchors; full-resolution vertices are deterministic after the rigid projection. We train RigidFormer on NVIDIA A100 GPUs for 300 epochs with AdamW [23] (β_1=0.9, β_2=0.999, weight decay 0.01) at a base learning rate of 10^{-4}. The schedule combines a 10-epoch linear warmup (from 10% of the base rate) with cosine decay to $\eta_{\min}$=10^{-6}, and we clip the gradient norm at 1.0 for stability. Additional implementation details are provided in Appendices C and G.

4 Experiments

We conduct experiments to evaluate RigidFormer: (i) *Accuracy* vs. state-of-the-art learning-based simulators; (ii) *Generalization* across datasets; (iii) *Resolution generalization* to unseen test-time point counts; (iv) *Anchor robustness* to varying anchor counts and anchor perturbations (random sampling); (v) *Ablations* of anchor-based 3D RoPE, gated attention, and differentiable rigid projection; (vi) *Scalability and controllability* for large scenes and articulated bodies; (vii) *Runtime performance*; and (viii) *Partial point clouds*. Please refer to the *project page* for more qualitative results.

Datasets & Metrics. We use the MOVi (Multi-Object Video) datasets [15]: *MOVi-A* (basic geometric shapes), *MOVi-B* (complex geometric shapes), and *MOVi-Sphere* (spheres). Following [1, 33, 39], we report Translation RMSE (m) for per-object center-of-mass position and Orientation RMSE (deg) via quaternion geodesic distance. We evaluate at physical frames during autoregressive rollout, following the HopNet protocol [39]. For our variable step sizes (1/5/10), predictions are mapped to the corresponding physical frames for fair comparison. All metrics are averaged over the test set. We use MOVi-B for ablations due to its diversity and vertex-level variation (see statistics in Appendix D).

4.1 Main Comparison

Tab. 2 reports the matched step-size-1 setting on MOVi benchmarks. We compare RigidFormer (point input, †) against mesh-based simulators (e.g., HopNet, FIGNet, MGN), point-based methods (VPD), and Transformer-based approaches (HCMT). Despite using no mesh connectivity, RigidFormer obtains the best position and orientation errors in every reported column. On MOVi-B at 100 frames, RigidFormer improves over HopNet from 0.176/ 17.91° to 0.108/ 12.65°, while using only point

Table 2: **Standard Performance on MOVi-A, MOVi-B, and MOVi-Sphere.** Each cell reports position RMSE (m) / orientation RMSE (°) at prediction horizons of 50, 75, and 100 frames. Per-column top-three ranks are denoted by shading: 1st, 2nd, 3rd.

Model	MOVi-A			MOVi-B			MOVi-Sphere		
	50	75	100	50	75	100	50	75	100
MGN+	0.705 / 31.21	N/A	N/A	0.538 / 26.91	N/A	N/A	N/A	N/A	N/A
MGN-LargeRadius+	0.119 / 15.07	N/A	N/A	0.460 / 26.34	N/A	N/A	N/A	N/A	N/A
FIGNet	0.115 / 14.84	N/A	N/A	0.127 / 13.99	N/A	N/A	N/A	N/A	N/A
FIGNet$_{reimpl}$	0.132 / 7.10	0.285 / 14.62	0.492 / 23.30	0.141 / 7.39	0.300 / 15.16	0.516 / 24.96	N/A	N/A	N/A
HCMT$_{reimpl}$*	0.239 / 5.70	0.538 / 11.82	0.951 / 18.40	0.237 / 4.72	0.527 / 9.80	0.932 / 17.43	0.243 / 4.19	0.541 / 8.21	0.956 / 13.81
VPD$^{\dagger}_{reimpl}$	0.235 / 5.10	0.489 / 11.66	0.827 / 20.37	0.275 / 4.65	0.581 / 9.70	0.987 / 16.99	0.244 / 4.47	0.510 / 9.50	0.855 / 17.65
HopNet	0.054 / 5.64	0.115 / 11.84	0.196 / 18.83	0.047 / 4.91	0.101 / 10.35	0.176 / 17.91	0.034 / 4.05	0.073 / 8.21	0.124 / 13.68
RigidFormer †,*	0.040 / 4.81	0.086 / 10.64	0.145 / 17.57	0.030 / 3.29	0.061 / 7.16	0.108 / 12.65	0.022 / 2.93	0.045 / 6.28	0.078 / 10.74

∗ Transformer-based model; † Point inputs.

Table 3: **Generalization performance across MOVi-Sphere (S), MOVi-A (A), and MOVi-B (B).** The left block compares FIGNet, HopNet, and RigidFormer under the matched step-size-1 setting, with top-three rank shading: 1st, 2nd, 3rd. The right block reports RigidFormer with larger step sizes; step size 10 predicts only frames that are multiples of 10 (–).

Train	Test	FIGNet		HopNet		RigidFormer step=1	
		75	100	75	100	75	100
S	A	0.370 / 14.08	0.626 / 22.46	0.112 / 11.29	0.197 / 17.74	0.122 / 10.79	0.205 / 17.57
	B	0.662 / 18.08	1.149 / 29.66	0.106 / 9.75	0.188 / 17.13	0.081 / 8.56	0.138 / 15.02
A	S	0.239 / 11.55	0.417 / 19.66	0.100 / 9.16	0.172 / 15.17	0.059 / 7.17	0.100 / 12.59
	B	0.512 / 13.49	0.864 / 22.41	0.117 / 10.77	0.202 / 18.66	0.074 / 8.07	0.132 / 14.46
B	S	0.536 / 12.74	0.905 / 20.42	0.095 / 9.09	0.160 / 15.04	0.063 / 7.78	0.110 / 13.43
	A	0.520 / 14.53	0.871 / 22.89	0.120 / 12.08	0.202 / 19.15	0.099 / 10.62	0.165 / 17.18

Train	Test	RigidFormer step=5		RigidFormer step=10	
		75	100	75	100
S	A	0.099 / 8.32	0.178 / 14.80	–	0.139 / 11.37
	B	0.067 / 7.10	0.123 / 13.31	–	0.094 / 10.04
A	S	0.046 / 5.46	0.083 / 10.34	–	0.074 / 7.58
	B	0.057 / 6.66	0.108 / 12.40	–	0.085 / 9.77
B	S	0.053 / 6.24	0.095 / 11.64	–	0.082 / 8.67
	A	0.078 / 8.20	0.134 / 14.37	–	0.108 / 11.63

inputs. For fairness, the input points are the mesh vertices themselves, as used by mesh-based methods, even when sparse for simple shapes (e.g., 51 points for a cube, 64 for a cone); Sec. 4.4 uses partial point clouds. We run VPD [40] with at most 1,536 points per scene (Appendix C). The SDF-Sim [33] code is not publicly available, so we cannot rerun it under our full protocol; in the MOVi-B @50 result reported by Rubanova et al., SDF-Sim obtains 0.160/ 18.03° (their FIGNet: 0.140/ 15.00°), whereas RigidFormer obtains 0.030/ 3.29° at matched step size 1 and 0.012/ 1.22° at step size 10, without SDF fitting. Qualitative results are provided in Fig. 2 and on the project page. We evaluate temporal discretization through step-size-conditioned rollouts in Tab. 5.

Cross-Dataset Generalization. Tab. 3 reports cross-dataset transfer by training on one MOVi variant and testing on another. In the matched step-size-1 block, RigidFormer improves over FIGNet in every split and over HopNet in every split except the position error from MOVi-Sphere to MOVi-A. The larger-step block shows that step-size conditioning further reduces our long-horizon errors in every transfer split. Training on MOVi-Sphere transfers least well in position; we analyze this trend in Appendix D.1.

Zero-Shot Transfer to MOVi-C. Following HopNet [39], we evaluate the models trained on MOVi-Sphere, MOVi-A and MOVi-B without adaptation on the 120 MOVi-C test scenes of HopNet, which hold 3 to 10 scanned everyday objects [11] unlike the training data in shape, size and mass (Appendix H). At step size 1 and 50 frames, RigidFormer has 8–18% lower position errors than HopNet from all training sets (Tab. 4); its orientation error is lower from MOVi-B, nearly equal from MOVi-A and higher from MOVi-Sphere. Step size 10 lowers its errors further.

Table 4: **MOVi-C, zero-shot**: position (m) / orientation (°) RMSE at 50 frames. HopNet as reported in [39]. Step size 10: given frames spaced by the step.

Trained on	Step size 1		Step size 10
	HopNet	RigidFormer	RigidFormer
MOVi-Sphere	0.104 / 6.56	0.096 / 7.55	0.032 / 2.26
MOVi-A	0.109 / 6.96	0.095 / 6.95	0.032 / 2.12
MOVi-B	0.105 / 7.09	0.086 / 5.88	0.032 / 2.02

Point Resolution Generalization. Tab. 6 reports point-cloud results on MOVi-B. RigidFormer is trained with randomly sampled point counts $\{128, 256, 512, 1024\}$ and evaluated at 768 points. Despite the unseen test-time resolution, the model remains stable across 50/75/100-frame rollouts[1]; at 100 frames, step sizes 10, 5, and 1 obtain 0.137/11.13°, 0.161/14.83°, and 0.189/16.22°, respectively.

Step Sizes. We investigate the effect of step-size conditioning in Tab. 5. Larger step sizes consistently yield better long-horizon performance because the model performs fewer autoregressive updates over

[1] In Tabs. 6–10, the 75-frame entries of step size 10 are evaluated at frame 80, the nearest frame on its grid.

Table 5: **Step-size-conditioned rollout performance.** Each cell reports position RMSE (m) / orientation RMSE (°). Step size 10 predicts only frames that are multiples of 10 (–).

Step Size	MOVi-A			MOVi-B			MOVi-Sphere		
	50	75	100	50	75	100	50	75	100
10	0.016 / 1.43	–	0.092 / 11.26	0.012 / 1.22	–	0.071 / 8.74	0.010 / 0.88	–	0.062 / 6.58
5	0.029 / 2.94	0.068 / 7.76	0.120 / 13.76	0.020 / 2.71	0.048 / 6.32	0.090 / 11.52	0.017 / 2.02	0.040 / 4.83	0.073 / 8.85
1	0.040 / 4.81	0.086 / 10.64	0.145 / 17.57	0.030 / 3.29	0.061 / 7.16	0.108 / 12.65	0.022 / 2.93	0.045 / 6.28	0.078 / 10.74

Table 6: **Generalization to different point resolutions.** Trained with point counts 128, 256, 512, 1024 and evaluated at 768 points. Each cell reports position RMSE (m) / orientation RMSE (°).

Step size	Rollout horizon (frames)		
	50	75	100
10	0.071 / 1.64	0.101 / 6.11	0.137 / 11.13
5	0.082 / 3.55	0.113 / 8.06	0.161 / 14.83
1	0.094 / 4.38	0.132 / 9.21	0.189 / 16.22

Table 7: **Effect of Gated Attention and Differentiable Rigid Alignment.** Top-three values shaded by rank among variants.

Gated Attention	Differentiable Alignment	Metric	step_size=1			step_size=5			step_size=10		
			50	75	100	50	75	100	50	75	100
✗	✓	Pos	0.077	0.153	0.259	0.050	0.107	0.191	0.033	0.090	0.152
		Ori	4.70	9.98	17.12	3.51	7.77	13.93	1.57	6.02	10.78
✓	✗	Pos	0.052	0.098	0.169	0.042	0.082	0.146	0.030	0.072	0.121
		Ori	4.08	8.79	15.79	3.14	7.39	13.75	1.55	5.98	11.04
✓	✓	Pos	0.050	0.095	0.161	0.040	0.078	0.136	0.029	0.069	0.115
		Ori	3.97	8.51	15.33	3.06	7.25	13.55	1.51	5.89	10.85

the same physical horizon. Step size $= 10$ gives the lowest 100-frame errors on MOVi-A, MOVi-B, and MOVi-Sphere. The step size $= 1$ rows remain the matched setting for comparison with prior one-step rollout protocols, while step size $= 5$ provides an intermediate trade-off between rollout frequency and local prediction difficulty. More discussion can be found in Appendix M.

4.2 Ablation Studies

Positional Embedding. We compare our Anchor-based Rotary Positional Embedding (ARoPE) with standard sinusoidal PE [36], learned absolute PE [10], and geometry-aware OBB, PCA, and SE(3) baselines. ARoPE uses the set-aggregated anchor descriptor from Sec. 3.3 to rotate query and key vectors as a function of sparse 3D anchor coordinates, while mean aggregation makes the descriptor invariant to anchor reindexing. As reported in Tab. 8, ARoPE achieves the best or tied-best position error in eight of nine cells and the best orientation error in most cells; SE(3) and sinusoidal PE are close in selected metrics. Overall, the results suggest that direct anchor-coordinate RoPE provides a stable positional signal for heterogeneous MOVi-B shapes.

Gated Attention. We ablate the sigmoid gating in RigidFormer. Tab. 7 shows that gating mainly improves translation accuracy and long-horizon stability. At 100 steps, it reduces position error from $0.259 \rightarrow 0.161$ (step_size$=1$), $0.191 \rightarrow 0.136$ (step_size$=5$), and $0.152 \rightarrow 0.115$ (step_size$=10$). Orientation error also improves in most cells; at step size 10 and 100 steps, it remains essentially tied with the ungated variant.

Differentiable Kabsch Alignment. Tab. 7 also compares RigidFormer **w/** vs. **w/o** differentiable rigid projection. With gating enabled, the differentiable variant consistently reduces 100-step position error ($0.169 \rightarrow 0.161$, $0.146 \rightarrow 0.136$, and $0.121 \rightarrow 0.115$ for step sizes 1, 5, and 10). Orientation error also decreases at the same horizons, indicating that gradient flow through the rigid projection reduces rollout drift.

Different Numbers of Anchors. Tab. 9 studies the effect of the anchor count k on MOVi-B. Increasing from $k = 3$ to $k = 4$ improves translation accuracy across step sizes and rollout horizons. Using $k = 8$ remains competitive and yields lower orientation error in several long-horizon settings, but it uses twice as many anchor queries and offers a less favorable translation–cost trade-off. We therefore use 4 anchors as the default efficiency–quality trade-off.

Table 8: **Comparison of Different Positional Embeddings.** Top-three values shaded by rank among PE variants.

PE	Metric	step_size=1			step_size=5			step_size=10		
		50	75	100	50	75	100	50	75	100
Sinusoidal	Pos	0.051	0.100	0.172	0.042	0.083	0.145	0.030	0.074	0.122
	Ori	4.10	8.68	15.30	3.16	7.39	13.52	1.55	6.04	10.91
Learned	Pos	0.054	0.103	0.176	0.042	0.082	0.146	0.030	0.073	0.122
	Ori	4.24	9.01	15.94	3.18	7.43	13.77	1.56	5.98	11.04
OBB	Pos	0.060	0.114	0.189	0.045	0.087	0.155	0.033	0.079	0.133
	Ori	4.38	9.44	16.83	3.31	7.69	14.35	1.57	5.96	11.08
PCA	Pos	0.380	0.867	1.548	0.240	0.647	1.249	0.107	0.499	0.918
	Ori	7.02	15.11	25.79	5.23	12.52	22.62	2.27	9.32	16.74
SE(3)	Pos	0.052	0.099	0.167	0.041	0.080	0.139	0.029	0.069	0.114
	Ori	4.25	9.21	16.36	3.15	7.45	13.89	1.52	5.97	11.04
ARoPE (Ours)	Pos	0.050	0.095	0.161	0.040	0.078	0.136	0.029	0.069	0.115
	Ori	3.97	8.51	15.33	3.06	7.25	13.55	1.51	5.89	10.85

Randomized FPS Anchors. We train and evaluate RigidFormer to assess whether the model learns *intrinsic geometric features* of the

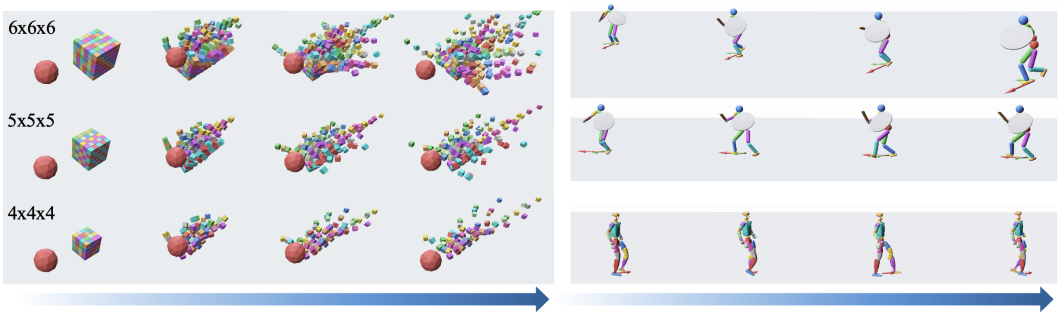


Figure 3: **Scalability and controllability. Left:** Simulation with $216, 125, 64$ cubes; RigidFormer stays stable as object count grows. **Right:** Preliminary direction-controlled articulated dynamics. Humanoid: commanded facing and moving directions are the green and red arrows, respectively; Unitree G1: the arrow shows the moving direction. RigidFormer follows heading commands while preserving coherent part-level motion.

Table 9: **Different Numbers of Anchors.** Results for different numbers of anchors in RigidFormer. Each cell reports position RMSE (m) / orientation RMSE (°). Top three shaded, also in Tab. 10.

Anchors	Step size	Rollout horizon (frames)		
		50	75	100
3	10	0.033 / 1.63	0.081 / 6.74	0.135 / 12.71
	5	0.047 / 3.63	0.094 / 8.67	0.166 / 16.80
	1	0.060 / 4.82	0.118 / 10.64	0.198 / 19.01
4	10	0.029 / 1.51	0.069 / 5.89	0.115 / 10.85
	5	0.040 / 3.06	0.078 / 7.25	0.136 / 13.55
	1	0.050 / 3.97	0.095 / 8.51	0.161 / 15.33
8	10	0.031 / 1.51	0.079 / 5.53	0.130 / 10.07
	5	0.046 / 3.14	0.097 / 7.20	0.170 / 13.38
	1	0.061 / 4.08	0.123 / 8.67	0.208 / 15.26

Table 10: **Randomized FPS anchors.** We train and evaluate with FPS anchors randomly re-sampled as a perturbation study. Each cell reports position RMSE (m) / orientation RMSE (°).

Anchors	Step size	Rollout horizon (frames)		
		50	75	100
3	10	0.031 / 1.72	0.075 / 6.36	0.126 / 11.74
	5	0.042 / 3.72	0.085 / 8.40	0.149 / 15.12
	1	0.055 / 4.80	0.110 / 10.46	0.187 / 18.19
4	10	0.029 / 1.59	0.070 / 5.76	0.115 / 10.39
	5	0.041 / 3.21	0.081 / 7.23	0.139 / 13.38
	1	0.052 / 4.41	0.099 / 9.31	0.163 / 16.16
8	10	0.030 / 1.52	0.083 / 5.58	0.141 / 10.06
	5	0.044 / 3.23	0.095 / 7.28	0.171 / 13.29
	1	0.056 / 4.04	0.114 / 8.64	0.193 / 15.17

object, rather than relying on specific anchor identities. Tab. 10 reports a perturbation study where FPS anchors are randomly re-sampled during both training and evaluation; it is separate from the default fixed-N_a=4 evaluation used elsewhere. Randomized FPS with 4 anchors gives the strongest translation accuracy, while 8 anchors can reduce rotation error in several settings. These results indicate that RigidFormer is robust to anchor selection and that the default N_a=4 provides a good quality–efficiency trade-off.

4.3 Scalability, Controllability and Efficiency

Large-Scale Simulation. We demonstrate the scalability of RigidFormer on the WreckingBall dataset, which contains scenes with 64, 125, and 216 cubes (visualized in Fig. 3, left). At 50 physical steps, RigidFormer achieves position RMSEs of 1.210, 0.690, and 0.130, with orientation errors of $20.50°$, $16.50°$, and $4.60°$ for the three scenes, respectively. The configurations differ in more than object count, so they are stress tests of one model rather than an object-count ablation (App. J); the model remains stable in all of them. In this setting, RigidFormer runs at 20 FPS.

Controllable Articulated-Body Simulation. We further evaluate RigidFormer as a preliminary extension to articulated characters under directional control. For the ASE humanoid with 17 body parts and the Unitree G1 robot with 31 body parts, we treat each body part as an object-level component and condition the model on desired heading commands through FiLM. As shown in Fig. 3, RigidFormer produces coherent whole-body motion that follows different heading commands, suggesting that the object-centric formulation can extend beyond independent rigid objects. At 100 physical steps, RigidFormer obtains position/orientation errors of 0.062 / $14.47°$ on ASE Humanoid and 0.072 / $16.26°$ on G1. We note that this result is presented as an extension study rather than the primary evidence for the method.

Runtime Performance. Tab. 11 reports runtime on MOVi-B using a 50-step autoregressive rollout averaged over 10 iterations on an NVIDIA GeForce RTX 5080. RigidFormer achieves $28\times$ and $354\times$ speedups over FIGNet and HopNet, respectively. This efficiency mainly comes from the object-centric Transformer design,

Table 11: Runtime comparison.

Method	ms/step	FPS
HopNet	4228.7	0.2
FIGNet	336.0	3.0
RigidFormer (Ours)	**12.0**	**83.6**

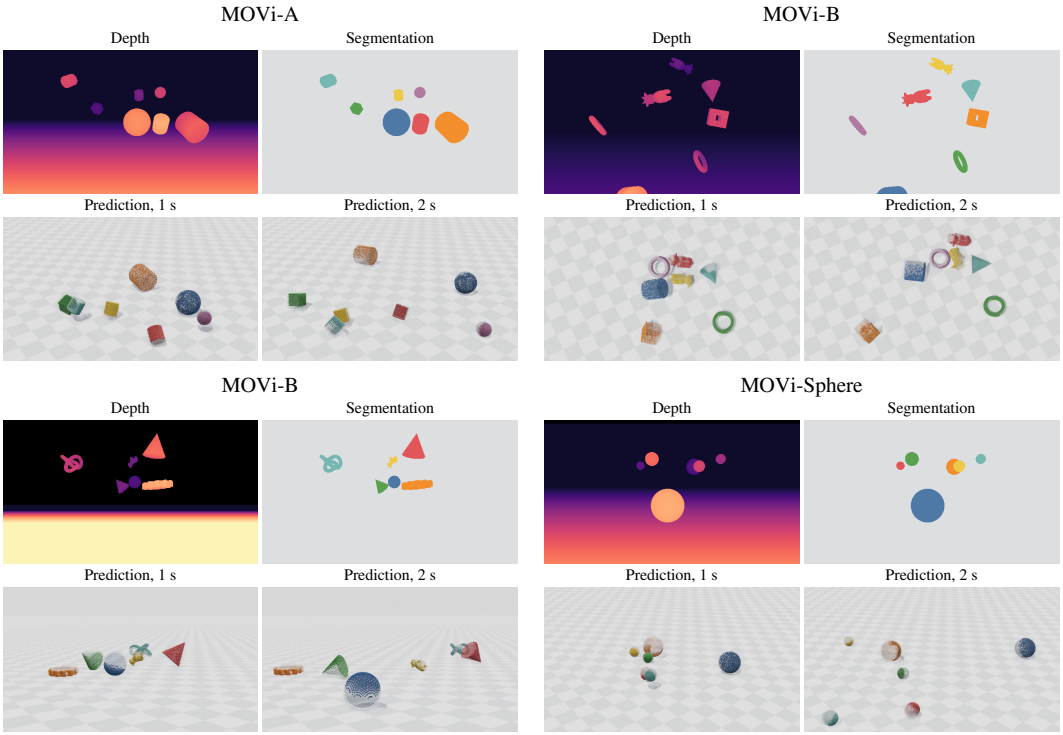


Figure 4: **Partial point clouds.** Test scenes of MOVi-A, MOVi-B and MOVi-Sphere observed by a single depth camera. For each scene: the depth and segmentation images (the inputs) and the predicted rollout after 1 s and 2 s. Opaque points are observed; light points are completed.

which reasons over compact object-level states rather than dense vertex-level structures. Further runtime analysis is provided in Appendix F.1.

4.4 Partial Point Clouds

RigidFormer also simulates scenes observed by a single depth camera (Fig. 4). The camera observes each scene once, so each object is given as a partial point cloud: the back-projected points of its segment, which cover only the side facing the camera. The model takes these points either directly or after a point completion network fills in the occluded side; the architecture is unchanged, the points take the place of the mesh vertices, and the model is trained on these inputs. Appendix I describes the camera, the completion network and the training details, and compares both inputs (Tab. I17).

5 Conclusion

We presented RigidFormer, an object-centric, mesh-free Transformer-based simulator for multi-object rigid-body contact dynamics from point clouds. RigidFormer shifts interaction reasoning to the *object level* and updates dynamics using a compact set of anchors, reducing vertex-level computation while maintaining accuracy. To encode the objects' spatial layout, we introduce Anchor-based RoPE, whose mean-pooled anchor descriptor is invariant to anchor reindexing and whose use for object tokens preserves permutation equivariance over unordered objects. RigidFormer further enforces rigidity by projecting updates onto the rigid-body manifold via differentiable Kabsch alignment, and it supports controllable integration through step-size conditioning within a single model. Experiments validate the efficiency, effectiveness, and versatility of RigidFormer, including zero-shot transfer to scanned objects and simulation from single-camera partial point clouds. *Limitations and future work.* The current formulation primarily targets object-level rigid-body contact dynamics and relies on object labels to identify which points belong to each object. Although we report partial point-cloud results, prediction becomes challenging when partial observations capture too little of an object's shape. Future work could move toward noisier perception settings, mixed rigid–deformable scenes, and more fine-grained adaptive time stepping (Appendix N).

Acknowledgments and Disclosure of Funding

This work was conducted within the framework of the collaboration between MIT and Meta and was supported by funding from MIT and Meta.

References

[1] Kelsey R Allen, Yulia Rubanova, Tatiana Lopez-Guevara, William Whitney, Alvaro Sanchez-Gonzalez, Peter Battaglia, and Tobias Pfaff. Learning rigid dynamics with face interaction graph networks. *arXiv preprint arXiv:2212.03574*, 2022.

[2] Genesis Authors. Genesis: A universal and generative physics engine for robotics and beyond. `https://github.com/Genesis-Embodied-AI/Genesis`, 2024.

[3] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[4] Romain Brégier. Deep regression on manifolds: a 3D rotation case study. In *2021 International Conference on 3D Vision (3DV)*, pages 166–174. IEEE, 2021.

[5] Arunkumar Byravan and Dieter Fox. SE3-Nets: Learning rigid body motion using deep neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 173–180. IEEE, 2017. doi: 10.1109/ICRA.2017.7989023.

[6] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.

[7] Hsiao-yu Chen, Edith Tretschk, Tuur Stuyck, Petr Kadlecek, Ladislav Kavan, Etienne Vouga, and Christoph Lassner. Virtual elastic objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15827–15837, 2022.

[8] Ricky TQ Chen, Brandon Amos, and Maximilian Nickel. Learning neural event functions for ordinary differential equations. *arXiv preprint arXiv:2011.03902*, 2020.

[9] Erwin Coumans and Yunfei Bai. PyBullet, a Python module for physics simulation for games, robotics and machine learning, 2016.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[11] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B. McHugh, and Vincent Vanhoucke. Google Scanned Objects: A high-quality dataset of 3D scanned household items. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2553–2560, 2022.

[12] C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

[13] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[14] Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38:293–306, 1985.

[15] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J. Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora,

Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3749–3761, June 2022.

[16] Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer. In *European Conference on Computer Vision*, pages 289–305. Springer, 2024.

[17] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. DiffTaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*, 2019.

[18] Wenlong Huang, Yu-Wei Chao, Arsalan Mousavian, Ming-Yu Liu, Dieter Fox, Kaichun Mo, and Li Fei-Fei. PointWorld: Scaling 3D world models for in-the-wild robotic manipulation. *arXiv preprint arXiv:2601.03782*, 2026.

[19] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923, 1976. doi: 10.1107/S0567739476001873.

[20] Chanho Kim and Fuxin Li. Object dynamics modeling with hierarchical point cloud-based representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20977–20986, 2024.

[21] Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, Yutian Chen, Junjie Yan, Ming Wei, Y. Zhang, Fanqing Meng, Chao Hong, Xiaotong Xie, Shaowei Liu, Enzhe Lu, Yunpeng Tai, Yanru Chen, Xin Men, Haiqing Guo, Y. Charles, Haoyu Lu, Lin Sui, Jinguo Zhu, Zaida Zhou, Weiran He, Weixiao Huang, Xinran Xu, Yuzhi Wang, Guokun Lai, Yulun Du, Yuxin Wu, Zhilin Yang, and Xinyu Zhou. Attention residuals, 2026. URL `https://arxiv.org/abs/2603.15031`. arXiv:2603.15031.

[22] Jiahui Lei, Yijia Weng, Adam W Harley, Leonidas Guibas, and Kostas Daniilidis. MoSca: Dynamic Gaussian fusion from casual videos via 4D motion scaffolds. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6165–6177, 2025.

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[24] Miles Macklin. Warp: A high-performance Python framework for GPU simulation and graphics. `https://github.com/nvidia/warp`, March 2022. NVIDIA GPU Technology Conference (GTC).

[25] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac Gym: High performance GPU-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[26] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (ToG)*, 40(4):1–20, 2021.

[27] Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. ASE: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions On Graphics (TOG)*, 41(4):1–17, 2022.

[28] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[29] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International conference on learning representations*, 2021.

[30] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[31] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[32] Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, et al. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *arXiv preprint arXiv:2505.06708*, 2025.

[33] Yulia Rubanova, Tatiana Lopez-Guevara, Kelsey R Allen, William F Whitney, Kimberly Stachenfeld, and Tobias Pfaff. Learning rigid-body simulators over implicit shapes for large-scale scenes and vision. *Advances in Neural Information Processing Systems*, 37:125809–125838, 2024.

[34] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[35] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[37] Chen Wang, Roberto Martín-Martín, Danfei Xu, Jun Lv, Cewu Lu, Li Fei-Fei, Silvio Savarese, and Yuke Zhu. 6-PACK: Category-level 6D pose tracker with anchor-based keypoints. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10059–10066. IEEE, 2020. doi: 10.1109/ICRA40945.2020.9196679.

[38] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19795–19806, 2023.

[39] Amaury Wei and Olga Fink. Integrating physics and topology in neural networks for learning rigid body dynamics. *Nature Communications*, 16(1):6867, 2025.

[40] William F Whitney, Tatiana Lopez-Guevara, Tobias Pfaff, Yulia Rubanova, Thomas Kipf, Kimberly Stachenfeld, and Kelsey R Allen. Learning 3D particle-based simulators from RGB-D videos. *arXiv preprint arXiv:2312.05359*, 2023.

[41] William F Whitney, Jacob Varley, Deepali Jain, Krzysztof Choromanski, Sumeet Singh, and Vikas Sindhwani. Modeling the real world with high-density visual particle dynamics. *arXiv preprint arXiv:2406.19800*, 2024.

[42] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. PointFlow: 3D point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4541–4550, 2019.

[43] Youn-Yeol Yu, Jeongwhan Choi, Woojin Cho, Kookjin Lee, Nayong Kim, Kiseok Chang, Chang-Seung Woo, Ilho Kim, Seok-Woo Lee, Joon-Young Yang, et al. Learning flexible body collision dynamics with hierarchical contact mesh transformer. *arXiv preprint arXiv:2312.12467*, 2023.

[44] Jingyang Yuan, Gongbo Sun, Zhiping Xiao, Hang Zhou, Xiao Luo, Junyu Luo, Yusheng Zhao, Wei Ju, and Ming Zhang. EGODE: An event-attended graph ODE framework for modeling rigid dynamics. *Advances in Neural Information Processing Systems*, 37:59093–59118, 2024.

[45] Chong Zeng, Yue Dong, Pieter Peers, Hongzhi Wu, and Xin Tong. RenderFormer: Transformer-based neural rendering of triangle meshes with global illumination. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–11, 2025.

[46] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. LION: Latent point diffusion models for 3D shape generation. *Advances in Neural Information Processing Systems*, 35:10021–10039, 2022.

[47] Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. MonST3R: A simple approach for estimating geometry in the presence of motion. *arXiv preprint arXiv:2410.03825*, 2024.

[48] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021.

[49] Haoyu Zhen, Qiao Sun, Hongxin Zhang, Junyan Li, Siyuan Zhou, Yilun Du, and Chuang Gan. TesserAct: learning 4D embodied world models. *arXiv preprint arXiv:2504.20995*, 2025.

[50] Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Extending Lagrangian and Hamiltonian neural networks with differentiable contact models. *Advances in Neural Information Processing Systems*, 34:21910–21922, 2021.

[51] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5745–5753, 2019.

Appendix Contents

A. More Qualitative Results 16
B. Theoretical Properties of the Object-Anchor Representation 16
C. Training Details 19
D. Data Statistics 20
E. Data Augmentation 21
F. Runtime Performance and Computational Costs 21
G. Detailed Network Architecture 22
H. Zero-Shot Transfer to MOVi-C 22
I. Partial Point Clouds from a Single Depth Camera 23
J. Large-Scale Simulation 24
K. Controllable Articulated Body Simulation 25
L. Additional Ablations and Analyses 27
M. Discussion on Temporal Step Sizes 28
N. Limitations, Future Work, and Impact 28

A More Qualitative Results

More results on partial point-cloud inputs are shown in Fig. A5. We randomly mask 25% of the points inside each object's bounding box and evaluate the model from the resulting partial observations. Additional videos are included on the project page.

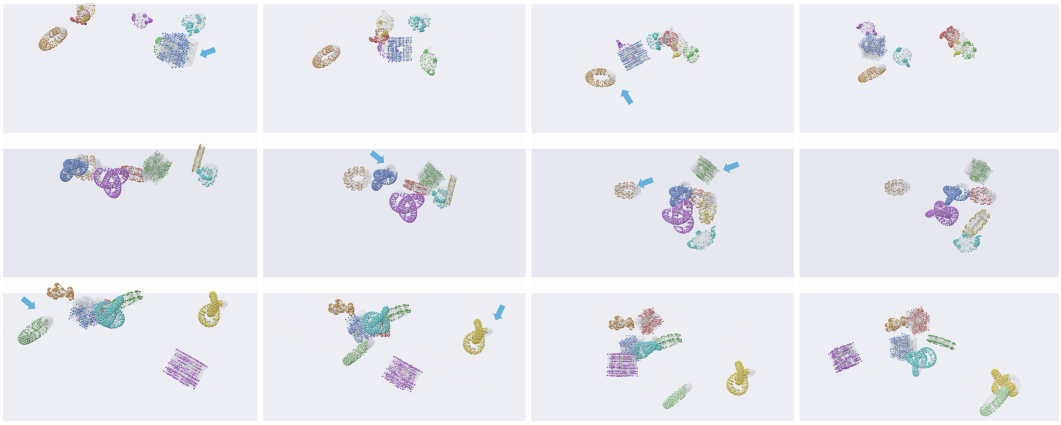


Figure A5: **Partial point-cloud qualitative results.** RigidFormer predicts accurate rigid-body interactions from partial observations with low drift and stable contacts. Meshes are shown only for visualization; our model operates on point inputs. Please refer to the project page for more results.

We provide additional qualitative visualizations in Figs. A6–A8, covering held-out MOVi-A, MOVi-B, and MOVi-Sphere test scenes. Each row corresponds to one selected sample and shows four uniformly sampled frames from the rollout. These examples complement Fig. 2 and show that RigidFormer produces stable multi-object rollouts across different object geometries and contact configurations.

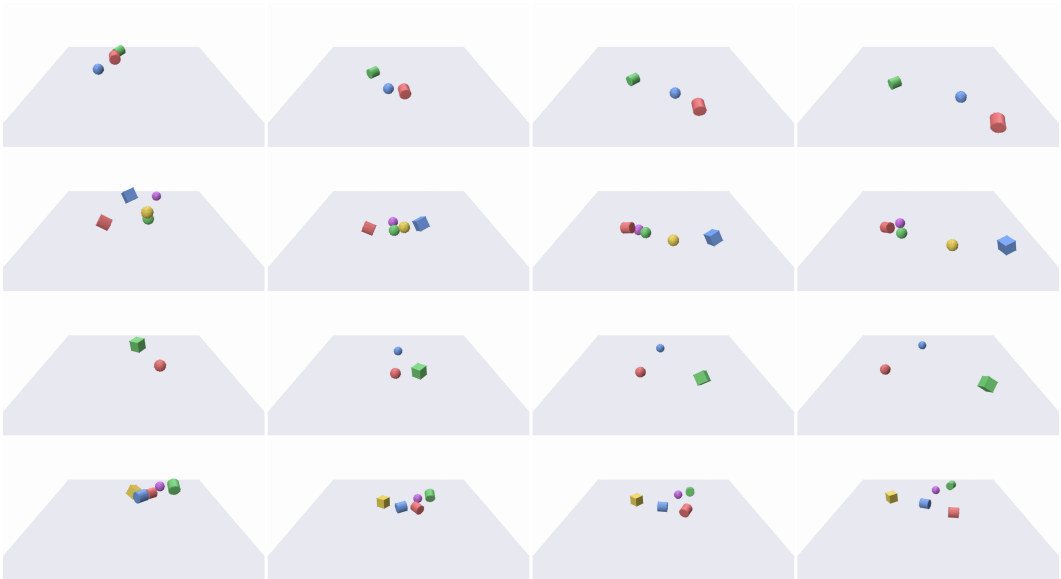


Figure A6: **More qualitative results on MOVi-A.** Each row is one held-out test sample; columns show four frames from the rollout.

B Theoretical Properties of the Object-Anchor Representation

We state the exact symmetry and structural guarantees used in Sec. 3.3. These properties concern indexing choices and rigid projection in the representation; they do not claim full physical equivariance of the simulator under arbitrary rigid transformations of the world.



Figure A7: **More qualitative results on MOVi-B.** Each row is one held-out test sample; columns show four frames from the rollout.

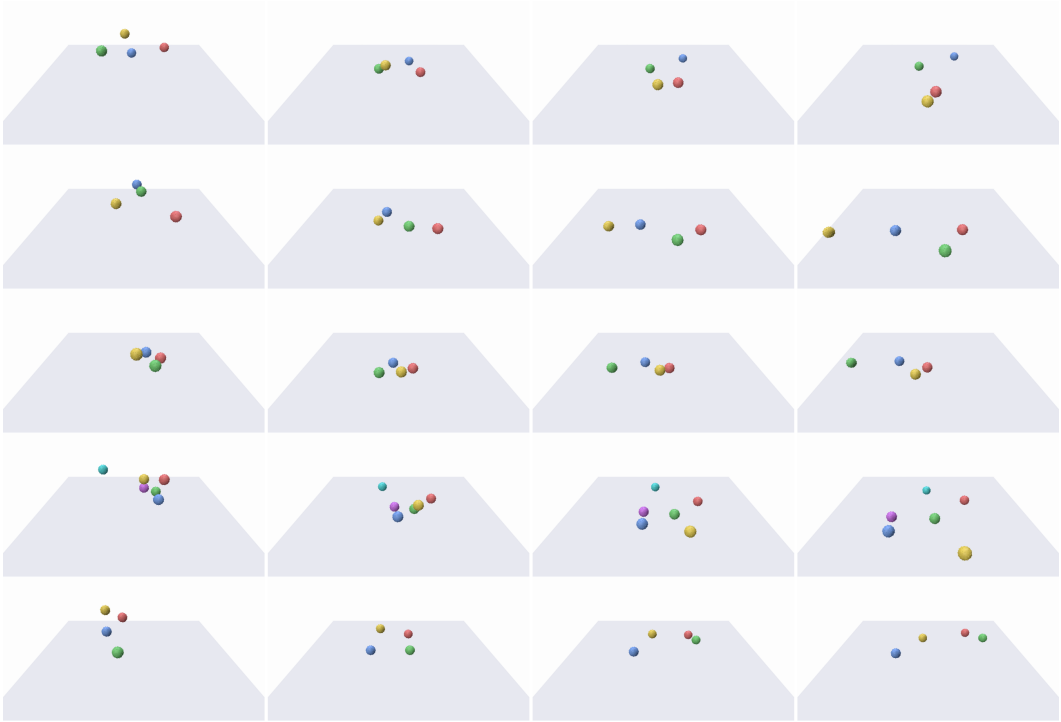


Figure A8: **More qualitative results on MOVi-Sphere.** Each row is one held-out test sample; columns show four frames from the rollout.

Anchor reindexing invariance. Let $\psi_\omega(\mathbf{x})$ denote the shared 3D rotary anchor map used by ARoPE, and define the object descriptor

$$\rho(A_i) = \frac{1}{N_a}\sum_{k=1}^{N_a}\psi_\omega(\mathbf{x}_k^{(i)}), \qquad A_i = \{\mathbf{x}_k^{(i)}\}_{k=1}^{N_a}. \tag{5}$$

For any permutation π of the anchors,

$$\rho(\{\mathbf{x}_{\pi(k)}^{(i)}\}_{k=1}^{N_a}) = \frac{1}{N_a}\sum_{k=1}^{N_a}\psi_\omega(\mathbf{x}_{\pi(k)}^{(i)}) = \frac{1}{N_a}\sum_{k=1}^{N_a}\psi_\omega(\mathbf{x}_k^{(i)}) = \rho(A_i). \tag{6}$$

Thus the ARoPE descriptor is invariant to anchor ordering. The same statement holds for any shared per-anchor map followed by a symmetric pooling operator; we use mean pooling for simplicity and scale stability. Since ψ_ω is linear in the position, $\rho(A_i) = \psi_\omega(\bar{\mathbf{x}}^{(i)})$ with $\bar{\mathbf{x}}^{(i)} = \frac{1}{N_a}\sum_k \mathbf{x}_k^{(i)}$: the descriptor equals the rotary angles of the anchor centroid.

Vertex-order invariance of AVP. For a fixed anchor $\mathbf{q}^{(i,k)}$, AVP computes weights

$$w^{(i,k,v)} = \exp\left(-\frac{\|\mathbf{x}^{(i,v)} - \mathbf{q}^{(i,k)}\|}{\sigma}\right), \qquad \mathbf{u}^{(i,k)} = \frac{\sum_v w^{(i,k,v)}\mathbf{f}^{(i,v)}}{\sum_v w^{(i,k,v)}}. \tag{7}$$

If the vertices and their encoder features are reindexed by any permutation τ, both the numerator and denominator are unchanged after relabeling the summation index. Therefore AVP is invariant to vertex ordering. The weights also depend only on Euclidean distances: for any common rigid transform $\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t}$ and $\mathbf{q}' = \mathbf{R}\mathbf{q} + \mathbf{t}$ with $\mathbf{R} \in SO(3)$, $\|\mathbf{x}' - \mathbf{q}'\| = \|\mathbf{x} - \mathbf{q}\|$. Thus the AVP attention weights are unchanged by common rigid transforms of an object's current point and anchor coordinates. In the batched implementation, weights on padded vertices are set to zero before normalization, which is equivalent to summing only over valid vertices.

Object permutation equivariance. Consider a permutation matrix P acting on the M object tokens, and let $\tilde{P} = \mathrm{diag}(P, I_{N_r})$ act on the concatenation of object tokens and the N_r=16 register tokens. Self-attention without sequence-index positional embeddings satisfies

$$\mathrm{Attn}(\tilde{P}\mathbf{Q}, \tilde{P}\mathbf{K}, \tilde{P}\mathbf{V}) = \tilde{P}\,\mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}), \tag{8}$$

because the row-wise softmax and value aggregation commute with simultaneous token permutation. Token-wise RMSNorm, FiLM, elementwise gated attention, and FFNs also commute with $\tilde{P}$ since their parameters are shared across object positions. Therefore each decoder block maps a permutation of object tokens to the same permutation of outputs, while the register tokens remain a shared permutation-invariant workspace. Applying the same argument layer by layer proves equivariance of the object-level decoder.

Anchor prediction and rigid projection. The anchor predictor uses shared weights for each object and applies cross-object attention to the permuted object-token set; hence permuting object order only permutes the predicted anchor updates by object. Within one object, Kabsch alignment depends on sums of corresponding centered source and target anchors. If source and target anchors are reindexed by the same permutation, the cross-covariance matrix is unchanged, so the recovered rigid transform $(\mathbf{R}^{(i)}, \mathbf{t}^{(i)})$ is unchanged. The full simulator is therefore equivariant to object reordering and invariant to anchor reindexing under consistent anchor correspondences.

Rigidity by construction. After Kabsch alignment, every output vertex is generated as

$$\mathbf{x}_{t+1}^{(i,v)} = \mathbf{R}^{(i)}\mathbf{x}_{\mathrm{ref}}^{(i,v)} + \mathbf{t}^{(i)}, \qquad \mathbf{R}^{(i)} \in SO(3). \tag{9}$$

For any two vertices u, v of the same object,

$$\|\mathbf{x}_{t+1}^{(i,u)} - \mathbf{x}_{t+1}^{(i,v)}\|_2 = \|\mathbf{R}^{(i)}(\mathbf{x}_{\mathrm{ref}}^{(i,u)} - \mathbf{x}_{\mathrm{ref}}^{(i,v)})\|_2 = \|\mathbf{x}_{\mathrm{ref}}^{(i,u)} - \mathbf{x}_{\mathrm{ref}}^{(i,v)}\|_2. \tag{10}$$

Thus intra-object distances are preserved exactly by the final projection, independent of the raw anchor accelerations predicted by the network. The simulator is intentionally not SE(3)-equivariant overall: gravity, ground contact, and evaluation coordinates are expressed in the world frame.

C Training Details

C.1 Training Schedule

The final model is trained directly with sequence length $T{=}8$ and random integration step sizes $\Delta t \in \{1, 5, 10\}$ sampled with near-uniform probabilities. The same FiLM-conditioned model is therefore exposed to short and long temporal discretizations throughout training. Note that we did not use curriculum learning or scheduled sampling in the reported MOVi experiments.

We optimize with AdamW [23] using a base learning rate of 10^{-4}, weight decay 0.01, $\beta_1{=}0.9$, and $\beta_2{=}0.999$. The learning rate follows a 10-epoch linear warmup from 10% of the base rate, followed by cosine decay to 10^{-6}. Gradients are clipped to norm 1.0. Training runs for 300 epochs with 126 (MOVi-B) or 128 (MOVi-A, MOVi-Sphere) training sequences per step. We use HopNet-style $960/120/120$ train/validation/test split files for each MOVi variant. For a fair comparison, we follow the evaluation of HopNet [39], which gives four ground-truth frames (frames 0–3) followed by an autoregressive rollout; at larger step sizes, the given frames are spaced by the step.

VPD. Our VPD baseline uses 512 points per object; a scene with more than 1,536 points is reduced to a fixed random subset of 1,536 points, so that training fits in the memory of one GPU.

C.2 Loss Implementation Details

All loss terms in Sec. 3.4 are implemented using the Smooth L1 Loss [13] (PyTorch `nn.SmoothL1Loss()` function[2]), applied elementwise at anchor locations; the implementation sums coordinate losses over x, y, z and averages over valid anchors, objects, and time steps. Concretely, for two tensors $\mathbf{u}$ and $\mathbf{v}$ of the same shape, the Smooth L1 loss is defined as

$$\ell_{\text{SL1}}(\mathbf{u}, \mathbf{v}) = \frac{1}{|\mathbf{u}|} \sum_j \begin{cases} \frac{1}{2}(u_j - v_j)^2, & \text{if } |u_j - v_j| < 1, \\ |u_j - v_j| - \frac{1}{2}, & \text{otherwise}, \end{cases} \tag{11}$$

which corresponds to the default setting of `nn.SmoothL1Loss()` in PyTorch. Compared with ℓ_2 loss, Smooth L1 is less sensitive to occasional large errors during autoregressive rollout, while remaining smooth near zero and thus stable for optimization.

In our training objective, both the position and acceleration losses are computed using Smooth L1 before and after rigid projection. All four terms are evaluated at anchor locations. Specifically, the raw position loss is

$$\mathcal{L}_{\text{pos}}^{\text{raw}} = \ell_{\text{SL1}}\left(\hat{\mathbf{q}}_{t+1}^{\text{raw}}, \mathbf{q}_{t+1}^{\text{gt}}\right), \tag{12}$$

and the rigid position loss is

$$\mathcal{L}_{\text{pos}}^{\text{rigid}} = \ell_{\text{SL1}}\left(\hat{\mathbf{q}}_{t+1}^{\text{rigid}}, \mathbf{q}_{t+1}^{\text{gt}}\right), \tag{13}$$

where $\hat{\mathbf{q}}_{t+1}^{\text{raw}}$ denotes the predicted anchor positions before Kabsch alignment, $\hat{\mathbf{q}}_{t+1}^{\text{rigid}}$ denotes the corresponding anchors after rigid projection, and $\mathbf{q}_{t+1}^{\text{gt}}$ is the ground-truth anchor state. Full-resolution vertices are supervised indirectly through the rigid transform induced by the anchor update.

Similarly, the raw and rigid acceleration losses are defined as

$$\mathcal{L}_{\text{acc}}^{\text{raw}} = \ell_{\text{SL1}}\left(\hat{\mathbf{a}}_t^{\text{raw}}, \mathbf{a}_t^{\text{gt}}\right), \qquad \mathcal{L}_{\text{acc}}^{\text{rigid}} = \ell_{\text{SL1}}\left(\hat{\mathbf{a}}_t^{\text{rigid}}, \mathbf{a}_t^{\text{gt}}\right), \tag{14}$$

where the ground-truth acceleration is computed from the trajectory using the same Verlet discretization,

$$\mathbf{a}_t^{\text{gt}} = \frac{\mathbf{q}_{t+1}^{\text{gt}} - 2\mathbf{q}_t^{\text{gt}} + \mathbf{q}_{t-1}^{\text{gt}}}{\Delta t^2}. \tag{15}$$

The final loss is therefore

$$\mathcal{L} = \lambda_{\text{pos}}\left(\mathcal{L}_{\text{pos}}^{\text{raw}} + \mathcal{L}_{\text{pos}}^{\text{rigid}}\right) + \lambda_{\text{acc}}\left(\mathcal{L}_{\text{acc}}^{\text{raw}} + \mathcal{L}_{\text{acc}}^{\text{rigid}}\right), \tag{16}$$

with $\lambda_{\text{pos}} = 10$ and $\lambda_{\text{acc}} = 1$ in the reported experiments. For multi-step training, residuals are normalized by Δt^2 before the Smooth L1 reduction, matching the acceleration scale used by Verlet integration.

[2] `https://docs.pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html`

D Data Statistics

As detailed in Tab. D12, the datasets vary in object complexity. **MOVi-A** features simple geometric primitives—cubes (51 vertices), cylinders (64 vertices), and spheres (64 vertices)—uniformly distributed across scenes. **MOVi-B** introduces 11 object categories with diverse mesh complexities ranging from 51 vertices (cube) to 1,142 vertices (torus_knot), including complex shapes such as gear (569 vertices), sponge (908 vertices), spot (682 vertices), and suzanne (523 vertices); all categories appear with approximately equal frequency ($\sim$9% each). **MOVi-Sphere** serves as a controlled baseline containing only spheres (64 vertices each). This diversity enables comprehensive evaluation across varying geometric complexities, from simple 51-vertex primitives to intricate 1,142-vertex meshes. Each of MOVi-A, MOVi-B and MOVi-Sphere contains 1,200 scenes simulated for 480 frames. **MOVi-C**, used only for the zero-shot evaluation (Tab. 4, Appendix H), consists of the 120 test scenes provided by HopNet [39], each with 3 to 10 scanned everyday objects from Google Scanned Objects [11]: 768 objects of 518 distinct assets with 46 to 1,908 vertices (mean 540), simulated for 480 frames.

Table D12: **Object Types and Vertex Counts in MOVi Benchmarks.**

Dataset	Object Name	Vertices	Ratio
MOVi-A	cube	51	33.6%
	cylinder	64	33.8%
	sphere	64	32.6%
MOVi-B	cone	64	8.8%
	cube	51	9.3%
	cylinder	64	8.8%
	gear	569	8.8%
	sphere	64	9.2%
	sponge	908	8.7%
	spot	682	9.7%
	suzanne	523	9.1%
	teapot	274	9.4%
	torus	647	8.9%
	torus_knot	1142	9.2%
MOVi-Sphere	sphere	64	100.0%
MOVi-C	518 scanned objects	46–1,908	$\leq$0.7% each

Table D13: **Dataset summary statistics**.

Dataset	Scenes	Total instances	Object types	Frames/scene
MOVi-A	1,200	7,810	3 primitives	480
MOVi-B	1,200	7,764	11 shapes	480
MOVi-Sphere	1,200	7,810	1 primitive	480
MOVi-C	120 (test)	768	518 scanned objects	480

D.1 Cross-Dataset Transfer Analysis

Tab. 3 shows that the MOVi-Sphere-trained model transfers least well to the other MOVi variants. Averaged over the two held-out target datasets, the 100-step position errors of the MOVi-Sphere-trained model are 0.172, 0.151, and 0.117 for step sizes 1, 5, and 10, respectively. The corresponding averages are 0.137, 0.115, and 0.095 for MOVi-B training, and 0.116, 0.095, and 0.080 for MOVi-A training. The gap is larger in translation than in orientation: from MOVi-Sphere to MOVi-A, the orientation error at 100 steps is close to that of MOVi-B training (17.57° against 17.18°), while the position error is 24% higher (0.205 against 0.165). One interpretation is related to geometry. MOVi-Sphere contains only spheres, whose contacts involve no edges or faces, whereas MOVi-A contains three primitives and MOVi-B 11 categories with much broader shape variation (Tab. D12); a model trained on spheres sees fewer of the contact configurations that occur in the other datasets.

This interpretation is meant only to explain the average trend; it is not evidence that cross-dataset transfer is generally easier than in-domain evaluation.

E Data Augmentation

To improve generalization and prevent overfitting, we employ two complementary data augmentation strategies during training:

Rotation Augmentation. We apply random Z-axis rotations to the entire scene during training. Given the original vertex positions $\mathbf{X} \in \mathbb{R}^{N\times 3}$, we apply:

$$\mathbf{X}' = \mathbf{X}\mathbf{R}_z(\theta)^\top \tag{17}$$

where $\mathbf{R}_z(\theta)$ is a rotation matrix around the Z-axis (gravity direction). The rotation angle θ is sampled from discrete values $\{5^\circ, 10^\circ, 15^\circ, \ldots, 355^\circ\}$ with angle step 5°. Rotation is sampled once per training batch: with probability 0.5, the whole batch is rotated by the same sampled angle. This improves robustness to yaw rotations around the gravity axis rather than imposing full 3D rotation invariance, which would be inappropriate for scenes with gravity and ground contact.

Importantly, the rotation is applied consistently to all objects in a scene and to all frames in a sequence, preserving the relative object relationships and temporal dynamics. Since the full trajectory is rotated before acceleration targets are computed, accelerations transform consistently:

$$\mathbf{a}' = \mathbf{R}_z(\theta)\mathbf{a} \tag{18}$$

Object Permutation Augmentation. Multi-object dynamics should be permutation-equivariant: the physics outcome should not depend on the arbitrary ordering of objects in the input. The architecture already satisfies this property for object tokens (Appendix B); during training, we additionally randomly permute object ordering with 50% probability as a robustness check. Given N_{obj} objects, we sample a random permutation $\pi \in S_{N_{\text{obj}}}$ and reorder all per-object tensors accordingly:

$$\{\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(N_{\text{obj}})}\} \to \{\mathbf{X}^{(\pi(1))}, \ldots, \mathbf{X}^{(\pi(N_{\text{obj}}))}\} \tag{19}$$

This augmentation is applied in the data loader before collation and feature computation, ensuring consistent permutation across all object attributes (positions, velocities, physics parameters). It helps detect implementation mistakes that would otherwise introduce order dependence and improves robustness to arbitrary input arrangements at inference time.

Rotation is applied after moving each training batch to the GPU, while object permutation is applied on the data-loading side before collation. Both add negligible overhead to the training pipeline.

F Runtime Performance and Computational Costs

F.1 Runtime Performance

We profile inference on MOVi-B scene 72, containing 10 objects with a total of $4{,}016$ vertices, using a 50-step rollout setting. Per-object vertex counts are $\{64, 51, 1142, 64, 64, 682, 682, 647, 51, 569\}$, covering diverse MOVi-B meshes including torus_knot ($1{,}142$), spot (682), torus (647), gear (569), cube (51), and several primitives (64 vertices each). As summarized in Tab. F14, the model uses 107.0M parameters at inference (148.4M including disabled optional modules) and has a peak memory footprint of 0.82 GB.

We time each part with CUDA events at its boundaries; the events add about 5% to the step, so the parts sum to 12.43 ms instead of the 11.96 ms of Tab. 11. The model core (point encoder + object-level interaction + anchor-object interaction) takes 9.99 ms per step (80.3%). Within the model core, the point encoder takes 1.78 ms (14.3%), object-level interaction takes 2.46 ms (19.8%), and anchor-object interaction takes 5.75 ms (46.2%). Geometric processing takes 1.12 ms per step (9.0%): the CUDA KNN search 0.43 ms (3.5%) and the differentiable Kabsch transform 0.69 ms (5.6%). The remaining 1.32 ms (10.6%) covers integration, ground-plane handling, anchor-vertex pooling and bookkeeping. These results suggest that further speedups should primarily target the anchor-object interaction.

Table F14: RigidFormer inference performance on the MOVi-B evaluation set (times with CUDA events at the boundaries of each part; 11.96 ms per step, 83.6 FPS, without them).

Overall Performance		
Parameters (inference / total)	107.0M / 148.4M	
Peak Memory	0.82 GB	
Total Objects	10	
Total Vertices	4016	
Model Core Components		
Component	Time (ms)	Percent
Point Encoder	1.78	14.3%
Object-Level Interaction	2.46	19.8%
Anchor-Object Interaction	5.75	46.2%
Subtotal	**9.99**	**80.3%**
Geometric Processing		
CUDA KNN	0.43	3.5%
Diff Kabsch Transform	0.69	5.6%
Subtotal	**1.12**	**9.0%**
Other (integration, ground plane, pooling, bookkeeping)	1.32	10.6%
Total	**12.43**	**100%**

Vertex-Level Processing Analysis. To illustrate the efficiency of our object-level tokenization, we analyze the hypothetical computational cost if attention were computed at the vertex level. With $N_v = 4{,}016$ vertices, a single self-attention layer would require $4 \times N_v^2 \times D = 4 \times 4016^2 \times 768 \approx 49.5$ GFLOPs. In contrast, our object-level decoder self-attends over 10 object tokens and 16 unpositioned register tokens, requiring $4 \times 26^2 \times 768 \approx 2.1$ MFLOPs per layer for the attention matrix-value product. This is a $\sim 2.4 \times 10^4$ reduction in the quadratic attention term before accounting for linear projections and FFNs. The measured runtime in Tab. F14 includes the full model and geometric processing; the main cost is the anchor-object interaction rather than object-level attention.

G Detailed Network Architecture

Tab. G15 summarizes the architecture used in the main experiments. The model uses 107.0M parameters at inference and consists of a hierarchical PointNet encoder, a 4-layer object-level Transformer decoder with gated attention and register tokens, and a query-based anchor predictor. The publicly available weights additionally contain 41.4M parameters of optional modules that are disabled in all reported experiments (148.4M in total).

H Zero-Shot Transfer to MOVi-C

Data. MOVi-C [15] scenes hold 3 to 10 objects from Google Scanned Objects [11]. We use the 120 test scenes provided by HopNet [39] (480 frames at 240 frames per second; 768 objects of 518 assets) in the format of our MOVi-A, MOVi-B and MOVi-Sphere data: each object is represented by the vertices of its collision mesh (46 to 1,908 vertices, mean 540, without decimation), posed by the recorded positions and orientations. Kubric scales an asset by the recorded scale divided by the largest extent of the asset's bounding box, so that its largest dimension equals the recorded scale; with this factor (1.9 to 58.9), the meshes reproduce the recorded 3D bounding boxes to a relative error of 10^{-7}, and every resting object lies 2 mm above the floor, as in our other data. The provided metadata list the objects that are visible in the video; in 10 scenes, 11 simulated objects are never visible and are therefore not included. We use the scenes as provided, as HopNet does.

Physical parameters. The model reads the recorded masses, friction and restitution. The masses equal the volumes of the unscaled assets (10^{-5} to 0.036) and lie below every training mass (at least

Table G15: **Architecture Overview.** Summary of model components and their configurations.

Component	Configuration	Notes
Input Configuration		
Max objects per scene	$M = 16$	–
Vertices per object	all, no limit	up to 1,142 in training, 1,908 in MOVi-C
FPS anchors per object	$N_a = 4$	–
Encoder input dim	12D	–
– Displacement vector	$\mathbf{d} \in \mathbb{R}^3$	nearest contact geometry
– Velocity	$\mathbf{v} \in \mathbb{R}^3$	$\mathbf{x}_t - \mathbf{x}_{t-1}$
– Relative position	$\mathbf{r} \in \mathbb{R}^3$	$\mathbf{x}_t - \mathbf{x}_{\text{ref}}$
– Physics params	$[m, \mu, \epsilon] \in \mathbb{R}^3$	mass, friction, restitution
Time step	$[s, s^2] \in \mathbb{R}^2$	per-layer FiLM conditioning
Encoder (Hierarchical PointNet)		
Backbone convolutions	Conv1d MLP	shared over objects
Hierarchical pooling	4 levels (100%, 50%, 25%, 12.5%)	multi-scale geometry
Object embedding	$D = 768$	fixed-size object token
Decoder (4-Layer Transformer)		
Hidden dimension	$D = 768$	–
Attention heads	$H = 6$ (head dim = 128)	–
Object self-attention	Gated SDPA	elementwise sigmoid gate
Register tokens	16 learnable tokens	unpositioned global workspace
Time conditioning	per-layer FiLM	code $[s, s^2]$
FFN	SwiGLU, $2.5\times$ expansion	RMSNorm + QK norm
Block attention residuals	block size S=4	inter-block residual [21]
Dropout	0.1	attention and FFN
Anchor Predictor		
Anchor queries	N_a per object	FPS anchors
Anchor-Vertex Pooling	256D feature	learned isotropic distance kernel
Predictor input	271D	anchor state + AVP/context features
Cross-attention	object and cross-object context	anchor-level prediction
Output	3D anchor acceleration	followed by Verlet + Kabsch
Position Encoding and Projection		
ARoPE dimension	96	set-aggregated anchor RoPE
Rigid projection	Kabsch via RoMa	differentiable alignment
Total Parameters	107.0M at inference	148.4M incl. disabled modules

0.1); friction and restitution (0.5 for every object) lie between the training values; and 313 of the 768 objects are larger than any training object.

Protocol. The models of Tab. 2 are applied without any change and evaluated as in Sec. 4: four given frames (frames 0 to 3 at step size 1, spaced by the step otherwise), position and orientation errors per object, averaged over the 120 scenes. HopNet's values are those reported in [39] (zero-shot, 50 frames). Tab. H16 reports our errors up to 100 frames.

Table H16: **Zero-shot transfer to MOVi-C**: RigidFormer trained on each MOVi dataset, at step sizes 1, 5 and 10. Each cell reports position RMSE (m) / orientation RMSE (°); step size 10 predicts only frames that are multiples of 10 (–).

		Rollout horizon (frames)			
Trained on	**Step size**	25	50	75	100
MOVi-Sphere	1	0.026 / 2.02	0.096 / 7.55	0.212 / 15.68	0.367 / 25.58
	5	0.008 / 0.62	0.061 / 4.34	0.160 / 10.82	0.298 / 19.13
	10	–	0.032 / 2.26	–	0.240 / 15.78
MOVi-A	1	0.026 / 1.93	0.095 / 6.95	0.204 / 14.61	0.350 / 24.64
	5	0.008 / 0.58	0.058 / 4.01	0.149 / 9.97	0.272 / 17.77
	10	–	0.032 / 2.12	–	0.228 / 15.28
MOVi-B	1	0.021 / 1.60	0.086 / 5.88	0.187 / 12.25	0.320 / 20.38
	5	0.007 / 0.51	0.055 / 3.56	0.147 / 9.13	0.272 / 16.58
	10	–	0.032 / 2.02	–	0.229 / 14.63

I Partial Point Clouds from a Single Depth Camera

Inputs. A pinhole depth camera (640×480 pixels, 58° vertical field of view) observes each test scene once, at frame 3, from 12 m along the viewing direction of the scene's camera, aimed at a point

1.5 m above the center of the arena. Each object is represented by the back-projected points of its segment in its own frame, subsampled at random to at most 2,048 points and then to at most 1,024 by farthest point sampling (FPS). Objects with fewer than 16 points are left out.

Completion network. A PointNet encoder with a multilayer perceptron decoder takes the first 256 observed points of an object, the camera elevation and distance, and the height of the points above the ground, and predicts 512 points on the object's surface; FPS reduces the observed and predicted points to at most 1,024 (40–45% predicted). It is trained on the 960 MOVi-B training scenes and 2,400 further scenes from the MOVi-B shape library, which includes the shapes of MOVi-A and MOVi-Sphere, and then kept fixed.

Model. The architecture is unchanged: the observed points, or the observed and completed points, in FPS order, take the place of the mesh vertices, and the first four are the anchors. The model is given frames 0 to 3, with the points of frame 3 placed in frames 0 to 3 by the ground-truth poses; the camera is not used afterwards. The rigid motion of each object moves all of its points and, for evaluation, its collision mesh, which is scored as in the main protocol over the observed objects.

Training. Starting from the weights in Tab. 2, we train the model of each dataset on partial point clouds with one recipe: each sample is observed once by a camera like the evaluation camera at a random azimuth and completed by the fixed network, and the losses use the ground-truth positions of all points. We update all weights and the batch-normalization statistics with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 0.01 on weight matrices), a learning rate of 4×10^{-6} ($10\times$ for the fusion layer of the acceleration head), a 2-epoch linear warmup from $0.1\times$, a cosine decay to 5×10^{-8} over 60 epochs, gradient clipping at 1.0 and 21 scenes per step, and evaluate the exponential moving average of the weights (decay 0.9999). Without completion, we train only the encoder and the decoder, keeping the acceleration head and the batch-normalization statistics of Tab. 2, with the same schedule and an additional linear warmup from zero over the first epoch.

Visualization. Fig. 4 shows frames 240 and 476 (1 s and 1.98 s at 240 frames per second, step size 1). Observed points are opaque in their object's color, completed points smaller and lighter, and the mesh, moved by the same rigid motion, translucent. The depth and segmentation images are 640×360 center crops of the inputs.

Table I17: **Partial point clouds** (test scenes, step size 1): position RMSE (m) / orientation RMSE (°) at 50, 75 and 100 frames. Full: the mesh vertices, as in Tab. 2. Partial: the observed points of each object only. Partial + completion: the observed points with those of the completion network. Tab. 2: the models of Tab. 2, trained on full point clouds and applied unchanged. Trained: these models trained further on the given input (Training above).

		MOVi-A			MOVi-B			MOVi-Sphere		
Input	**Model**	50	75	100	50	75	100	50	75	100
Full	Tab. 2	0.040 / 4.81	0.086 / 10.64	0.145 / 17.57	0.030 / 3.29	0.061 / 7.16	0.108 / 12.65	0.022 / 2.93	0.045 / 6.28	0.078 / 10.74
Partial	Tab. 2	0.055 / 4.84	0.111 / 10.69	0.182 / 17.84	0.041 / 3.75	0.083 / 8.04	0.142 / 14.29	0.053 / 3.30	0.110 / 7.15	0.184 / 12.65
Partial	trained	0.055 / 4.82	0.110 / 10.52	0.182 / 17.53	0.040 / 3.67	0.080 / 7.90	0.136 / 13.90	0.053 / 3.26	0.108 / 7.11	0.180 / 12.60
Partial + completion	Tab. 2	0.052 / 5.24	0.097 / 10.73	0.162 / 18.02	0.036 / 3.75	0.075 / 7.88	0.129 / 13.48	0.023 / 3.08	0.048 / 6.54	0.083 / 11.10
Partial + completion	trained	0.047 / 4.71	0.092 / 9.78	0.156 / 16.44	0.034 / 3.54	0.069 / 7.54	0.120 / 13.21	0.026 / 2.92	0.054 / 6.23	0.093 / 10.63

J Large-Scale Simulation

To evaluate the scalability of RigidFormer to scenes with significantly more objects than the MOVi benchmarks, we construct the WreckingBall dataset featuring dense multi-object collision scenarios.

J.1 Dataset Generation

We simulate a wrecking ball scenario where a spherical projectile collides with a wall of stacked cubes. Four scene configurations are generated with varying grid sizes:

- **WreckingBall-28**: $3 \times 3 \times 3$ cube arrangement (M=27 cubes + 1 ball = 28 objects)
- **WreckingBall-65**: $4 \times 4 \times 4$ cube arrangement (M=64 cubes + 1 ball = 65 objects)
- **WreckingBall-126**: $5 \times 5 \times 5$ cube arrangement (M=125 cubes + 1 ball = 126 objects)
- **WreckingBall-217**: $6 \times 6 \times 6$ cube arrangement (M=216 cubes + 1 ball = 217 objects)

Data Generation Parameters. For each grid configuration, we generate 20 independent simulation samples, yielding a total of 80 samples across all configurations. The dataset is split into 72 training samples (18 per grid size) and 8 test samples (2 per grid size). Each trajectory spans T=600 frames at native simulation frequency. Cubes are placed in a deterministic regular grid with 1.05 m spacing. Per-sample variation comes from a randomized vertical offset of the ball's launch position (sampled uniformly from $[2.5, 4.0]$ m); the launch velocity is fixed at 30 m/s along the $+x$ axis toward the cube wall.

Reading the Four Configurations. The configurations differ in more than the number of cubes. The projectile has the same size and launch speed in all of them, while the wall grows from $3\times3\times3$ to $6\times6\times6$ cubes; the size of the projectile relative to the wall, the height of impact relative to the wall, the energy delivered per cube and the density of contacts therefore change together with the cube count. The four configurations are thus stress tests of the same model, not a controlled ablation of the object count.

Output Data Format. Each sample contains the following per-frame data:

- **Vertex positions**: Per-object vertex positions $\mathbf{X}_t^{(i)} \in \mathbb{R}^{N_v^{(i)}\times3}$ for each frame.
- **Mesh connectivity**: Static face connectivity stored per object for visualization only.
- **FPS anchors**: Deterministic Farthest Point Sampling anchors; the reported model uses N_a=4 anchors per object.

Physics parameters are consistent across all configurations: cube mass m=1.0 kg, friction coefficient μ=0.5, and restitution ϵ=0.3. Each cube mesh contains 8 vertices (corner points), yielding total vertex counts of 224 (28×8), 520 (65×8), 1008 (126×8), and 1736 (217×8) for the four configurations, respectively.

J.2 Training Configuration

We train RigidFormer on all four WreckingBall configurations jointly, using the mixed dataset of 72 training samples. The model uses the same object-anchor architecture as in the main MOVi experiments, including FiLM conditioning on $[s, s^2]$. It uses N_a=4 FPS anchors per object, and the maximum number of objects per frame is set to 220 to accommodate the largest configuration (217 objects) with a buffer. Training uses a step size s=10 for temporal prediction, with a sequence length of 8 frames per training sample. The batch size is set to 3 due to the large object count. We do not use curriculum learning in this experiment.

The object-level tokenization enables efficient scaling: attention complexity grows as $O(M^2)$ in object count rather than $O((MN_v)^2)$ in vertex count. For WreckingBall-217 with 1736 total vertices, vertex-level attention would require $\sim$3M attention pairs, while our object-level design requires only $\sim$54K pairs (217 objects + 16 registers), a $55\times$ reduction.

K Controllable Articulated Body Simulation

To provide a preliminary test of RigidFormer beyond independent rigid objects, we construct two articulated-body datasets with high-level control inputs. We treat each body part as an object-level component and condition the model on command signals.

K.1 Dataset Generation

ASE HumanoidHeading. We generate training data from the Adversarial Skill Embeddings (ASE) [27] framework using the HumanoidHeading task. The humanoid character is equipped with a sword and shield, comprising M=17 articulated body parts (15 body links plus the sword and shield). Data generation employs a hierarchical reinforcement learning architecture with pre-trained checkpoints: a Low-Level Controller (LLC) that generates joint torques $\boldsymbol{\tau} \in \mathbb{R}^{N_{\text{dof}}}$ from latent skill codes, and a High-Level Controller (HLC) that outputs latent skill codes conditioned on heading objectives.

Each trajectory spans $T{=}300$ frames at 30 Hz (10 seconds of simulation). Per-frame control signals $\mathbf{c}_t \in \mathbb{R}^5$ include:

- Target speed $v_{\text{tar}} \in \mathbb{R}$ (locomotion velocity magnitude)
- Target direction $\mathbf{d}_{\text{tar}} \in \mathbb{R}^2$ (movement direction unit vector)
- Target facing direction $\mathbf{f}_{\text{tar}} \in \mathbb{R}^2$ (heading orientation unit vector)

Meshes are simplified to a maximum of 300 vertices per body part using quadric edge collapse decimation. Mesh connectivity is used only for visualization; the model consumes point positions and per-point features, without surface normals.

G1 Steering. We generate data from the MimicKit framework using the Unitree G1 Steering task. The G1 robot comprises $M{=}31$ articulated body parts with significantly more complex geometry than ASE humanoids. Data generation uses the Adversarial Motion Priors (AMP) [26] steering policy trained for directional locomotion.

Each trajectory spans $T{=}300$ frames at 30 Hz. Per-frame control signals $\mathbf{c}_t \in \mathbb{R}^3$ include:

- Target speed $v_{\text{tar}} \in \mathbb{R}$ (locomotion velocity magnitude)
- Target direction $\mathbf{d}_{\text{tar}} \in \mathbb{R}^2$ (movement direction unit vector)

For G1, each body-part object is represented by 200 points sampled from the original robot mesh, without surface normals. Joint positions $\mathbf{q} \in \mathbb{R}^{N_{\text{dof}}}$ are stored for kinematic analysis.

Table K18: **Articulated Body Dataset Statistics.**

Property	**ASE Heading**	**G1 Steering**
Source Framework	ASE [27]	MimicKit (AMP)
Character	Humanoid + Sword/Shield	Unitree G1
Body Parts (M)	17	31
Max Vertices/Part	300	200
Control Dim	5	3
Frames (T)	300	300
Frequency	30 Hz	30 Hz

K.2 Implementation Details

Control Signal Integration. To condition RigidFormer on high-level control signals, we use the same FiLM mechanism as temporal step-size conditioning. The per-vertex encoder input remains 12D; the FiLM conditioning vector is augmented from $[s, s^2]$ to $[s, s^2, \mathbf{c}_t]$, where $\mathbf{c}_t \in \mathbb{R}^5$ for ASE and $\mathbf{c}_t \in \mathbb{R}^3$ for G1. Thus, control commands modulate the encoder/decoder features through FiLM rather than being concatenated to every point feature.

Mesh Processing Pipeline. Raw data export proceeds in two stages:

1. **Physics Export:** We run the trained policy in Isaac Gym [25] and record per-frame rigid-body transforms (position and quaternion) for each body part.
2. **Mesh Conversion:** Body part transforms are applied to reference meshes from MuJoCo XML files. Meshes exceeding vertex limits undergo quadric decimation.

Temporal Subsampling. We train with step size $s{=}10$ for both ASE and G1 (3 Hz effective rate) to capture meaningful locomotion dynamics rather than high-frequency contact oscillations.

Training Configuration. We use $N_a{=}4$ FPS anchors per body part and the same object-anchor architecture as in the main rigid-body experiments, including FiLM conditioning. The only additional conditions are the high-level control signals injected through FiLM. The batch size is 4 for both datasets. We present this result as an extension study rather than the primary evidence for the method.

L Additional Ablations and Analyses

L.1 Physical Parameters

Table L19: **Physical parameters of the objects** (all scenes of each dataset). Friction and restitution follow the material: 0.4 and 0.3 for metal, 0.8 and 0.7 for rubber, and 94% of the scenes of MOVi-A, MOVi-B and MOVi-Sphere hold both materials. The mass depends on the shape, the size and the material; in MOVi-C, it is the volume of the unscaled asset.

Dataset	Friction	Restitution	Mass	Distinct masses	Metal objects
MOVi-A	0.4 / 0.8	0.3 / 0.7	0.197–7.36	12	50.1%
MOVi-B	0.4 / 0.8	0.3 / 0.7	0.099–7.35	7,764 (all)	49.5%
MOVi-Sphere	0.4 / 0.8	0.3 / 0.7	0.197–3.87	4	50.8%
MOVi-C	0.5	0.5	1.0×10^{-5}–0.036	517 of 768	–

RigidFormer reads the mass, friction and restitution of each object (Sec. 3); Tab. L19 lists how they vary between objects. To measure how much it relies on them, we train on MOVi-B a model that gives every object the training-set means of the three parameters (m=1.12, μ=0.60, ϵ=0.50), with the architecture and training recipe of our model, and compare the two on the test scenes (Tab. L20: w/o and w/ physical parameter modeling). Without the physical parameters, the position error is 8 to 25% higher at step size 1 over frames 25 to 400 ($1.19\times$ at 100 frames, 95% interval of a paired bootstrap over scenes $[1.10, 1.31]$) and 8 to 31% higher at step sizes 5 and 10, while the orientation error changes by at most 7%: RigidFormer uses the parameters, mainly to place the objects.

Table L20: **Physical parameters** (MOVi-B test scenes): our model (w/ physical parameter modeling) and the model trained with the training-set means of the parameters for every object (w/o). Position RMSE (m) / orientation RMSE (°) at 50, 75 and 100 frames; step size 10 predicts no 75-frame entry here.

Step	Model	50	75	100
1	w/ physical parameter modeling	0.030 / 3.24	0.061 / 7.14	0.108 / 12.65
1	w/o physical parameter modeling	0.033 / 3.35	0.071 / 7.39	0.129 / 13.14
5	w/ physical parameter modeling	0.020 / 2.70	0.047 / 6.31	0.090 / 11.57
5	w/o physical parameter modeling	0.024 / 2.75	0.057 / 6.43	0.110 / 11.89
10	w/ physical parameter modeling	0.012 / 1.23	–	0.071 / 8.82
10	w/o physical parameter modeling	0.014 / 1.32	–	0.088 / 9.10

L.2 Encoder

We replace the hierarchical PointNet encoder (Sec. 3) with a set-abstraction encoder in the style of PointNet++ [31] (two set-abstraction levels with farthest point sampling and k-nearest-neighbor grouping, followed by feature propagation to all points) and train it on MOVi-B with the training recipe of our model. Tab. L21 compares it with our model on the test scenes. The set-abstraction encoder has a position error 1 to 13% higher at step size 1 over frames 25 to 400 ($1.06\times$ at 100 frames, 95% interval of a paired bootstrap over scenes $[0.97, 1.15]$), up to 22% higher at step size 5 and 3 to 9% higher at step size 10; its orientation error ranges from 1% lower to 18% higher. The set-abstraction encoder thus does not improve on the hierarchical encoder of RigidFormer.

Table L21: **Encoder** (MOVi-B test scenes): our model and the model with a set-abstraction encoder. Position RMSE (m) / orientation RMSE (°) at 50, 75 and 100 frames; step size 10 predicts no 75-frame entry here.

Step	Model	50	75	100
1	Our model	0.030 / 3.24	0.061 / 7.14	0.108 / 12.65
1	Set-abstraction encoder	0.032 / 3.76	0.065 / 7.91	0.114 / 13.66
5	Our model	0.020 / 2.70	0.047 / 6.31	0.090 / 11.57
5	Set-abstraction encoder	0.022 / 2.83	0.049 / 6.34	0.092 / 11.51
10	Our model	0.012 / 1.23	–	0.071 / 8.82
10	Set-abstraction encoder	0.012 / 1.28	–	0.077 / 8.74

L.3 Attention and Interaction Graphs

We relate the object-level self-attention of the decoder to contact. We roll out the models of Tab. 2 on the 120 test scenes of each dataset (step size 1) and average the attention between two object tokens over heads and both directions, at frames up to 100. Two objects are in contact when their meshes are closer than 1 cm, and never close when they stay farther than 10 cm within ± 5 frames.

In every decoder layer and on all three datasets, objects in contact receive more attention than objects that are never close (over the four layers $1.96\times$ on MOVi-B, $1.78\times$ on MOVi-Sphere and $1.28\times$ on MOVi-A; every 95% interval over scenes lies above 1), also at the same distance between their anchor centroids ($1.59\times$, $1.75\times$ and $1.22\times$). On MOVi-B, the attention grows as two objects approach (0.071 beyond 20 cm, 0.140 below 5 mm), and in the scene of Fig. L9 it rises within a few frames of each contact. These are correlations, and over all 476 rollout steps the ratio is lower (1.12 to 1.24).

M Discussion on Temporal Step Sizes

Step-size conditioning controls the effective temporal resolution of the learned simulator. A small step size is closest to standard one-step rollout protocols and returns dense future states, but a fixed physical horizon then requires many autoregressive calls, increasing both computation and accumulated rollout error. A larger step advances the learned world model over a longer interval per call, allowing it to expose sparse long-horizon future states more efficiently.

This trade-off is useful for planning. In many robotics settings, such as model-predictive planning or trajectory optimization, the planner needs to compare possible future object configurations and interaction outcomes, but does not always require every high-frequency contact frame. Coarse large-step rollouts can therefore provide a cheap long-horizon search signal, while smaller steps remain useful for local control, contact refinement, and short-horizon correction. A natural extension is adaptive time stepping: use large steps to explore candidate futures, then refine selected plans with smaller steps or uncertainty-aware rollouts.

N Limitations, Future Work, and Impact

N.1 Limitations and Future Work

RigidFormer is designed for object-level rigid-body dynamics from point clouds and assumes object labels that indicate which points belong to each object. Although we include partial point-cloud results, by masking points inside each object's bounding box (Appendix A) and from a single depth camera with completion (Appendix I), severe partial observations can become challenging when the visible points capture too little of the object's shape. Future work could study stronger occlusions, real sensor noise, online object segmentation from raw observations, mixed rigid–deformable scenes, and more fine-grained adaptive time stepping.

The rigid projection step exactly preserves intra-object distances, but contact handling is still learned from data rather than solved with an explicit complementarity-based physics engine. This design keeps the model mesh-free and efficient, while leaving room for contact-aware losses, lightweight correction layers, or hybrid learned-analytic constraints in scenes with unusually sharp contact

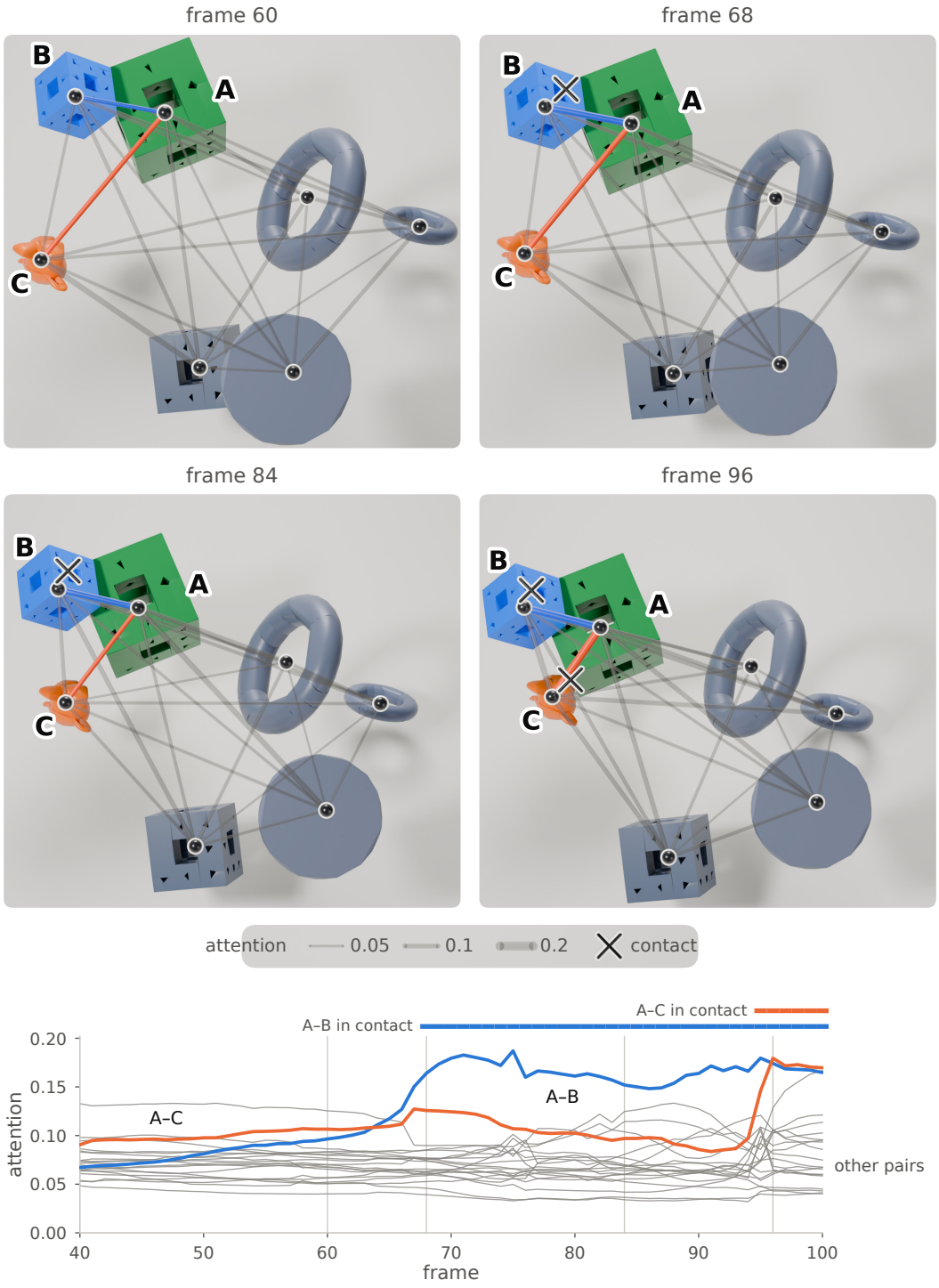


Figure L9: **Interaction graph of a MOVi-B test scene.** Top: the predicted states at four frames of the rollout (step size 1), rendered from one elevated view (objects A, B and C in green, blue and orange); each edge joins the anchor centroids of two objects and its width increases linearly with the attention between them, averaged over heads, layers and both directions; crosses mark where two objects touch. Bottom: attention between object A and objects B and C, and between all other pairs (gray), over frames 40 to 100; bars mark the frames at which the two objects are in contact in the predicted state. In the ground truth, A and B touch at frames 68–73 and A and C at frames 94–96; in the rollout, A and B remain in contact (overlapping) after frame 73.

regimes. Our current study also focuses primarily on rigid objects, with articulated bodies treated as collections of object-level parts; extending the same representation to mixed rigid–deformable scenes is a natural next step.

RigidFormer also takes the mass, friction and restitution of each object as inputs, and all reported results use the values of the simulator; Appendix L.1 measures the effect of withholding them. In practice these quantities would have to be estimated, for example by online system identification from the observed motion, which fits naturally into the object-centric state. The per-point velocities are differences between consecutive frames, so the inputs assume temporally consistent point samples of each object.

The errors grow with the rollout horizon (Tab. 5), and we give no bound on this growth; larger step sizes reduce it but do not predict the frames in between. Dense, sustained contact remains the most difficult regime. The runtimes we report are those of the model on one GPU; a deployed system adds the cost of perception, segmentation and point completion.

N.2 Impact Statement

This work aims to improve machine learning methods for physical dynamics modeling, especially mesh-free simulation of rigid-body contact from point clouds. Efficient learned simulation can benefit robotics, graphics, and embodied AI by reducing the cost of rollout prediction, supporting planning over object interactions, and enabling controllable physical reasoning in virtual environments.

The main deployment consideration is reliability under distribution shift. Models derived from this work should be validated under the intended geometry, material, and contact conditions before being used in safety-relevant control loops. In high-stakes settings, predictions should be paired with uncertainty checks, safety constraints, and human or system-level oversight. The method operates on geometric state data rather than personal data, so we do not anticipate direct privacy risks from the core technique.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state the mesh-free rigid-body simulation setting, assumptions, contributions, and scope. The main experiments in Sec. 4 and the limitations in Sec. 5 support and qualify these claims.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: Limitations and future work are discussed in Sec. 5 and Appendix N, including object-level inputs with temporally consistent point samples, learned contact handling, noisy observations, mixed rigid–deformable scenes, and adaptive time stepping.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes].

Justification: The scoped symmetry properties of ARoPE and object-token processing are stated in Sec. 3. Appendix B gives the assumptions and proofs, including the fact that the method is not fully SE(3)-equivariant.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes].

Justification: We disclose architecture, losses, optimizer and schedule, data splits, warmup and rollout protocol, and evaluation metrics in Secs. 3–4 and Appendices C, D, and G. The code, configuration files and trained models that reproduce the main MOVi results are public (next question).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: The code (MIT license) with configuration files and evaluation scripts, the trained models and the MOVi-Sphere, MOVi-A and MOVi-B data are public and linked from the project page in the abstract: code at `https://github.com/Frank-ZY-Dou/Dynamics-Modeling/tree/main/RigidFormer`, models at `https://huggingface.co/frankzydou/RigidFormer`, data at `https://huggingface.co/datasets/frankzydou/RigidFormer-MOVi`. The README gives the commands that reproduce the main MOVi results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: Training and test protocols, metrics, datasets, architecture, optimizer, schedule, step-size sampling, and rollout setup are specified in Secs. 3–4 and Appendices C, D, and G.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes].

Justification: We report fixed-test-set averaged RMSE metrics over the full evaluation splits, with consistent metrics and protocols across methods and ablations. Rank-shaded tables are used to make relative performance differences explicit.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: Sec. 3.4 and Appendix C specify the training GPU class, epochs, batch size, optimizer, and schedule. Sec. 4.1 and Appendix F.1 report inference hardware and runtime/profiling details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The work uses synthetic/simulation datasets and does not involve personal data, human subjects, or surveillance data. We are not aware of any deviation from the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: The Impact Statement discusses positive applications in robotics, graphics, and embodied AI, as well as reliability considerations for deployment under distribution shift.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The work does not release a high-risk foundation model, generative media model, scraped dataset, or human-data asset requiring special access safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes].

Justification: We credit the public datasets, simulators, and baseline methods used, including MOVi/Kubric and the articulated-control frameworks, through the corresponding citations. Our public code and data preserve the upstream license notices and terms of use.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes].

Justification: The paper documents the generated WreckingBall and articulated-body evaluation data in the appendix, including the generation process, splits, scene sizes, and control inputs. The assets we provide are public and include accompanying documentation.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A].

Justification: This work does not involve crowdsourcing, user studies, or human-subject data collection.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: This work does not involve human subjects or crowdsourcing, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes].

    Justification: LLMs were used for language polishing, auxiliary manuscript consistency checks, and development assistance for parts of the data-processing and visualization code. All such code was manually reviewed and validated by the authors; LLMs were not used as part of the proposed model, training objective, inference procedure, dataset content generation, experimental decision making, or scientific claims.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.