RiCo: Neural Simulation of Rigid-Body Interactions via Local Contact Reasoning

Anonymous authors
Paper under double-blind review

Abstract

Accurate simulation of rigid-body interactions is essential for predictive physical world models. Despite recent progress in modeling object dynamics, capturing how local contacts between surfaces shape object motion remains challenging. While end-to-end world models predict interactions across entire scenes or objects, in practice, rigid-body contact is inherently local, and only nearby surfaces can directly exchange contact forces. Motivated by this observation, we introduce Rigid-body Contact Reasoning (RiCo), which represents interactions between objects through sparse neighborhoods of contact surface points. RiCo combines each point's state with the relative geometry, motion, and physical properties of nearby surfaces, then reasons across the object's points to determine how these local contacts jointly affect its motion. By confining cross-object reasoning to nearby surfaces while propagating contact information within each rigid body, RiCo retains fine-grained interaction details without the cost of modeling every pair of scene points. Such properties enable RiCo a higher accuracy and contact fidelity. Experiments on MOVi-benchmark demonstrate that RiCo reduces 100-frame position and orientation errors by 31–35% and approximately 38%, respectively, compared with baselines. Moreover, RiCo achieves high contact fidelity, with ground-truth-relative penetration-time and mean-depth differences of 11.0% and 2.22 mm, respectively. RiCo further generalizes zero-shot from small-scale training scenarios to scenes containing 270 objects. Our real-world multi-ball collision experiments further provide preliminary evidence of sim-to-real transfer.

1 Introduction

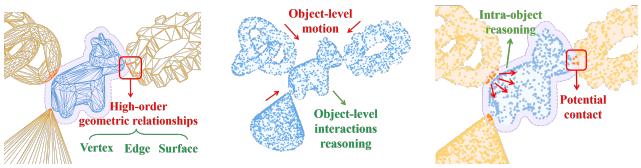
High-fidelity modeling of the physical world has become a central milestone in embodied AI, enabling agents to reason about how environments evolve and respond to their interactions (Agarwal et al., 2025). Recent advances in world models have learned predictive representations from large-scale video observations (Bruce et al., 2024; Ye et al., 2026; Li et al., 2026). Beyond these visual patterns, however, the physical world exhibits complex dynamics involving spatial motion, object geometry, and contact interactions (Hafner et al., 2023), which remain difficult to capture with video generation backbones. To advance the modeling of dynamic objects, we study neural simulation of rigid-body motions, including translation, rotation, and interactions among multiple objects in 3D space (Chang et al., 2016; Wei & Fink, 2025). As a fundamental abstraction of physical object motion, rigid-body dynamics exclude continuous deformation of objects, but introduce a significant challenge in modelling their interactions under different geometric form and motion conditions, because collisions introduce abrupt changes in linear and angular velocities, making contact dynamics non-smooth and challenging to predict accurately (Pfaff et al., 2020; Wei & Fink, 2025).

To address these challenges, prior works proposed learning neural approximations of motion dynamics from observed trajectories (Hafner et al., 2019; Brandstetter et al., 2022; Wu et al., 2023). Despite their success across various domains, modeling rigid-body dynamics remains challenging because collisions may induce abrupt changes in objects' velocities and subsequent motion trajectories. To better capture these discontinuities, recent studies incorporate geometric priors to explicitly model contact interactions. For instance, mesh-based neural simulators exploit mesh connectivity and surface geometry to propagate information about physical interactions (Pfaff et al., 2020; Allen et al., 2023). HOPNet (Wei & Fink, 2025) further incorporates higher-order topological structures to improve the modeling of rigid-body contact dynamics. However, processing mesh connectivity and constructing high-order geometric relationships can be computationally expensive and time-

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

consuming. Therefore, RigidFormer (Dou et al., 2026) represents individual objects using point sets and models their interactions through object-level attention, enabling rigid-body simulation without explicit mesh connectivity. While point clouds can efficiently capture surface geometry, they do not inherently encode which surface points may come into contact during object interactions.

Striving for efficient and fine-grained contact modeling, we propose RiCo which identifies and models local contact relationships directly from point-cloud representations, without relying on mesh connectivity. Figure 1 shows an illustrative example. This approach reflects the local nature of rigid-body interactions: only nearby surfaces can directly exchange contact forces (Sanchez-Gonzalez et al., 2020). Accordingly, instead of modeling the dense interactions among all objects, RiCo identifies only potential contact points and models their local dynamics across interacting objects. Because the combined effects of these contacts can govern rigid-body motion, we reason over contact information by propagating the effects of local contacts across rigid body. The intra-object contact reasoning provides a principled link between fine-grained interactions and whole-object motion. This design not only supports generalization across diverse shapes but also scales efficiently as the number of surface points grows.



(a) Mesh-based (b) RigidFormer (c) Ours

Figure 1: Comparing rigid body contact modeling.

To implement RiCo, we build a neural simulator that decomposes object-level collisions into sparse local contacts and intra-object contact reasoning. For each surface point, RiCo constructs a compact contact neighborhood containing nearby points from other objects and the environment. It then combines the source point's state with the relative geometry, motion, and physical properties of these contact candidates to form contact-conditioned point features. Based on such features, a point transformer attends each surface point to the others on the target object, so the model can reason where contacts occur, how the contacting surfaces move, and how forces at different locations affect translation and rotation. Based on these contact-conditioned features, an anchor-based decoder predicts motion updates at a small set of surface points, from which a single rigid transformation is recovered and applied to the entire object. This design integrates fine-grained geometric information into contact-dynamics reasoning, thereby generating physically plausible rigid-body motion.

We conduct extensive experiments on the MOVi benchmark (Greff et al., 2022) to evaluate RiCo in terms of prediction accuracy, generalization, contact fidelity, scalability, and real-world dynamics. Across diverse object configurations and long-horizon rollouts, RiCo reduces position and orientation errors by up to 35% and 38%, respectively. Beyond conventional trajectory metrics, we assess contact fidelity by comparing the predicted and ground-truth trajectories' aggregate penetration-time ratios and mean penetration depths, thereby accounting for geometric overlap already present in the ground-truth trajectories. Under this metric, RiCo reduces relative contact violations to 11.0% of object-time steps, with an average relative penetration depth of 2.22 mm. Moreover, RiCo generalizes effectively across unseen distributions of 270 objects. We further investigate zero-shot sim-to-real transfer through real-world multi-ball billiard collision experiments, providing preliminary evidence that the learned contact dynamics can generalize to real-world multi-object interactions without additional training. These results demonstrate that local contact reasoning enables accurate, contact fidelity, and scalable neural simulation of complex rigid-body dynamics.

2 Related Work

Learned Models of Physical Dynamics. Learned world models predict future environment states from observations, ranging from latent dynamics models (Ha & Schmidhuber, 2018; Hafner et al., 2019) to feature-space predictive approaches such as I-JEPA and DINO-WM (Assran et al., 2023; Zhou et al., 2024). Recent work adopts explicit 3D representations for embodied prediction: PointWorld (Huang et al., 2026) models action-conditioned scene evolution with 3D point flows, while RoboFlow4D (Lin et al., 2026) predicts multi-frame 3D motion fields for robotic manipulation. Despite this progress, rigid-body contact remains challenging because interactions can induce abrupt changes in global object motion, resulting in discontinuous dynamics that are difficult for smooth learned predictors to capture (Allen et al., 2023).

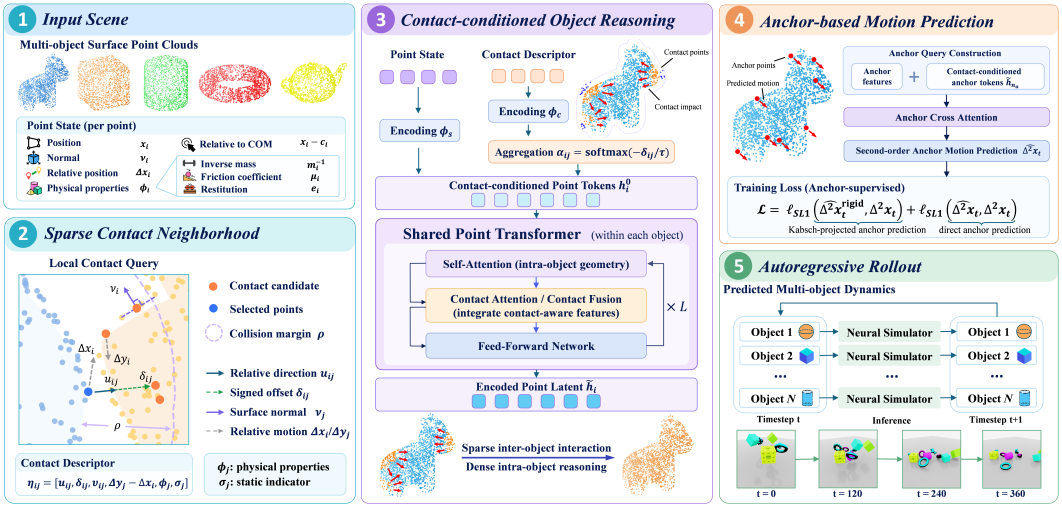


Figure 2: **Overview of RiCo.** From two consecutive scene states, RiCo constructs sparse local contact neighborhoods, fuses them with point-wise object states, and performs shared intra-object reasoning with a point transformer. The resulting contact-conditioned features are decoded at a small set of anchors to recover a rigid transformation for each object, and the predicted state is recursively used for autoregressive rollout.

Geometric Representations for Rigid-Body Simulation. Early learned physics models introduced inductive biases for predicting physical interactions (Battaglia et al., 2016; Sanchez-Gonzalez et al., 2018), which were later extended to particle-based simulators for objects and fluids (Li et al., 2018; Sanchez-Gonzalez et al., 2020). Mesh-based approaches further exploit explicit geometric structure by performing message passing over simulation meshes (Pfaff et al., 2020). For rigid-body contact dynamics, FIGNet (Allen et al., 2023) constructs interactions between mesh faces, preserving local surface geometry around contacts. Recent mesh-based models have explored hierarchical structures for propagating collision effects, while HOPNet (Wei & Fink, 2025) explicitly represents vertices, edges, mesh triangles, contacts, and objects through higher-order topological relations. To relax the dependence on meshes, SDF-Sim (Rubanova et al., 2024) represents object geometry using learned signed-distance functions, whereas RigidFormer (Dou et al., 2026) removes mesh connectivity entirely and operates on point-cloud inputs. Our method follows the mesh-free setting, while retaining point-level geometry for sparse local contact reasoning rather than compressing interactions primarily into coarse object-level representations.

3 Method

In this section, we formulate the problem of modeling rigid-body contact dynamics and introduce Rigid-body Contact Reasoning (RiCo), our neural simulator designed to solve the problem.

Problem Formulation. To model the rigid-body dynamics, we consider a scene containing M rigid objects, where M may vary across scenes. At time t, object i is represented by N_i sampled surface points $\mathbf{x}_t^i = [x_{t,1}^i, \ldots, x_{t,N_i}^i]^\top \in \mathbb{R}^{N_i \times 3}$, with corresponding surface normals ν_t^i, center of mass c_t^i, and physical properties $\boldsymbol{\phi}^i = [(m^i)^{-1}, \mu^i, e^i]$. We denote the full scene state by $\mathbf{s}_t = \{\boldsymbol{x}_t^i, \nu_t^i, c_t^i\}_{i=1}^M$. Given two consecutive states s_{t-1} and s_t, together with the static environment ξ, our goal is to build a dynamics model f_θ (i.e., state transition model) to predict $\hat{s}_{t+1} = f_\theta(s_{t-1}, s_t, \boldsymbol{\phi}, \xi)$. Appendix A introduces more details.

To effectively approximate f_θ, RiCo models the state update in three stages: (1) constructing a sparse representation of local contacts, (2) reasoning over contacts within each object, and (3) predicting rigid-body motion from anchors. Figure 2 provides an overview of the framework.

3.1 Sparse Contact Neighborhood Representation

Contact neighborhood construction. Striving for concise representation of contact dynamics, we construct a sparse local interaction representation around each surface point, rather than modeling interactions over complete mesh-level or object-level representations (Wei & Fink, 2025; Dou et al., 2026). Specifically, we first perform an object-level broad-phase query over active object pairs using

their surface-point AABBs, retaining only pairs whose minimum AABB separation is no greater than the contact radius ρ. More details can be found in appendix B.2. Candidates from all retained objects and the static environment ξ are then combined, from which the K nearest candidates within ρ are retained. Dynamic-object candidates therefore correspond to sampled surface points rather than closest points on continuous meshes, while static planar surfaces are queried analytically by projection. We denote the candidate positions by $\{y^i_{t,n,k}\}_{k=1}^K$[1].

To characterize the potential contact response, we gather each candidate's surface normal $\widetilde{\nu}^i_{t,n,k}$, finite-difference displacement $\Delta y^i_{t,n,k}$, and physical properties $\widetilde{\phi}^i_{t,n,k}$, comprising the inverse mass, friction coefficient, and coefficient of restitution of the corresponding external object or surface. Additionally, a binary indicator $\sigma^i_{t,n,k}$ distinguishes dynamic geometry ($\sigma = 0$) from static objects ($\sigma = 1$), with static candidates assigned zero displacement and zero inverse mass. We combine these quantities with the relative geometry and motion between the source point $x^i_{t,n}$ and each external candidate $y^i_{t,n,k}$ to form a 14-dimensional interaction descriptor:

$$\eta^i_{t,n,k} = \left[u^i_{t,n,k},\ \delta^i_{t,n,k},\ \widetilde{\nu}^i_{t,n,k},\ \Delta y^i_{t,n,k} - \Delta x^i_{t,n},\ \widetilde{\phi}^i_{t,n,k},\ \sigma^i_{t,n,k}\right] \in \mathbb{R}^{14}, \tag{1}$$

where $u^i_{t,n,k} = \frac{y^i_{t,n,k} - x^i_{t,n}}{\left\|y^i_{t,n,k} - x^i_{t,n}\right\|_2 + \varepsilon}$, $\delta^i_{t,n,k} = \left(x^i_{t,n} - y^i_{t,n,k}\right)^\top \widetilde{\nu}^i_{t,n,k}$.Within this formulation, $u^i_{t,n,k}$ encodes the direction from the source point to the candidate, with $\varepsilon > 0$ ensuring numerical stability. The relative displacement $\Delta y^i_{t,n,k} - \Delta x^i_{t,n}$ captures approach or separation and tangential sliding, while $\widetilde{\phi}^i_{t,n,k}$ specifies the physical properties of the external object. We also introduce an estimated signed distance $\delta^i_{t,n,k}$, which measures the signed offset of the source point from the local tangent plane on the candidate surface. Unlike the unsigned distance, its sign explicitly distinguishes the two sides of the local surface, providing a simple geometric cue to identify the potential penetration.

Point State Representation. To model how the source object responds to local interactions, we complement the contact descriptors with a point-wise state representation of its own geometry, motion, and physical properties. For point n on object i, we define this state as $s^i_{t,n} = \left[x^i_{t,n} - c^i_t,\ \Delta x^i_{t,n},\ \nu^i_{t,n},\ \phi^i,\ x^i_{t,n} - x^i_{0,n}\right] \in \mathbb{R}^{15}$. Within this representation, $x^i_{t,n} - c^i_t$ locates the point relative to the object's center of mass. The finite-difference displacement $\Delta x^i_{t,n} = x^i_{t,n} - x^i_{t-1,n}$ captures its motion over the preceding time step, while $\nu^i_{t,n}$ describes the local surface normal. The object-level properties $\phi^i = [(m^i)^{-1}, \mu^i, e^i]$ specify the inverse mass, friction coefficient, and restitution coefficient that govern contact response. Finally, $x^i_{t,n} - x^i_{0,n}$ records the point's displacement from its initial position, encoding its accumulated motion relative to the reference configuration.

Together, the point state $s^i_{t,n}$ and the local interaction descriptors $\{\eta^i_{t,n,k}\}_{k=1}^K$ capture the source point's own geometry, motion, and physical properties alongside those of nearby external surfaces.

3.2 Intra-Object Contact Reasoning

While contact neighborhoods identify potential interactions between nearby surfaces, predicting the resulting rigid-body motion requires reasoning jointly about contacts across the object's surface. We therefore fuse local contact descriptors with each point's state to form contact-conditioned tokens, then use a shared point Transformer to integrate these tokens within each object.

Contact Feature Aggregation. We embed the point state $s^i_{t,n}$ and each local interaction descriptor $\eta^i_{t,n,k}$ using MLPs ϕ_s and ϕ_c, respectively. To prioritize nearby surfaces while retaining information from multiple contact candidates, we aggregate the interaction embeddings $\phi_c(\eta^i_{t,n,k})$ using distance-dependent weights and fuse them with the point state embedding $\phi_s(s^i_{t,n})$. For the valid candidate set $\mathcal{C}^i_{t,n} \subseteq \{1, \ldots, K\}$, we define

$$h^{i,0}_{t,n} = \mathrm{RMSNorm}\left(\phi_s(s^i_{t,n}) + \sum_{k \in \mathcal{C}^i_{t,n}} \alpha^i_{t,n,k} \phi_c(\eta^i_{t,n,k})\right), \tag{2}$$

[1]If fewer than K valid candidates exist, the remaining entries are padded and masked during subsequent aggregation; when no valid candidate exists, the entire contact neighborhood is masked and contributes zero contact information.

where $\alpha^i_{t,n,k} = \frac{\exp(-d^i_{t,n,k}/\tau)}{\sum_{l\in\mathcal{C}^i_{t,n}}\exp(-d^i_{t,n,l}/\tau)}$, and $d^i_{t,n,k} = \|\mathbf{y}^i_{t,n,k} - \mathbf{x}^i_{t,n}\|_2$. Here, $d^i_{t,n,k}$ is the Euclidean distance, and $\tau > 0$ is a learnable temperature controlling the concentration of the weights [2].

Object-Wide Contact Integration. Predicting an object's response to contact requires reasoning about its interactions with surrounding objects and static surfaces. Although these interactions are encoded locally, their combined effect on rigid-body motion depends on contact conditions across the object's surface. As a result, for each object i, we collect the contact-conditioned point tokens as $\boldsymbol{h}^{i,0}_t = [h^{i,0}_{t,1}, \ldots, h^{i,0}_{t,N}]$ and process them independently using a shared B-block point Transformer $\mathcal{T}_\theta$. Each block performs gated multi-head self-attention followed by a feed-forward network. For a single attention head in block b, we compute:

$$\mathrm{Attn}_{\theta_b}(\boldsymbol{h}^{i,b}_t) = \left[\boldsymbol{g}^{i,b}_t \odot \mathrm{softmax}\left(\frac{(\boldsymbol{h}^{i,b}_t W^b_Q)(\boldsymbol{h}^{i,b}_t W^b_K)^\top}{\sqrt{d_{\mathrm{attn}}}}\right)\boldsymbol{h}^{i,b}_t W^b_V\right] W^b_O, \tag{3}$$

where $\boldsymbol{h}^{i,b}_t$ denotes the point tokens of object i at block b, and $\boldsymbol{g}^{i,b}_t = \sigma\left(\boldsymbol{h}^{i,b}_t W^b_G\right)$ is a learned sigmoid gate that modulates the attended features before output projection. W^b_Q, W^b_K, W^b_V, W^b_G, and W^b_O denote the query, key, value, gate, and output projections, respectively, and d_{attn} is the query/key dimension of each head. After B blocks, the resulting point representations are denoted by $\widetilde{\boldsymbol{h}}^i_t = \mathcal{T}_\theta(\boldsymbol{h}^{i,0}_t)$. The Transformer parameters are shared across objects, while all queries, keys, and values are derived exclusively from the point tokens of the same object. Importantly, since information from external surfaces is incorporated through the local contact fusion in Eq. 2, intra-object self-attention allows each point to access contact information from across its own object without dense cross-object attention. By restricting cross-object information exchange to sparse local contact neighborhoods, our method avoids dense point-level interactions across the entire scene.

3.3 Anchor-Based Rigid Motion Prediction and Training

Anchor-based Motion Prediction. As a contact-aware neural simulator, RiCo predicts each rigid object's future motion from its contact-conditioned tokens H^i_t. Since all surface points move under the same rigid transformation, predicting their future positions separately is often unnecessary. To preserve rigidity, we follow prior anchor-based formulations (Dou et al., 2026) and select A anchor samples for each object using deterministic farthest point sampling (FPS) (Qi et al., 2017). The corresponding sample indices $\{n^i_a\}_{a=1}^A$ remain fixed throughout the rollout, such that the anchors track the same surface samples over time. We predict motion updates only at these anchors and use them to recover the rigid transformation of the entire object.

$$\widehat{\Delta^2\boldsymbol{x}}^{\,i}_{t,\mathrm{anc}} = \mathcal{D}_\omega\left(\boldsymbol{z}^i_t, \widetilde{\boldsymbol{h}}^i_t\right) \text{ where } z^i_{t,a} = \widetilde{h}^i_{t,n^i_a} + \phi_q\left(\left[x^i_{t,n^i_a} - c^i_t,\ \Delta x^i_{t,n^i_a}\right]\right) \tag{4}$$

Here, $\boldsymbol{z}^i_t$ collects the A anchor queries, and $\mathcal{D}_\omega$ denotes the anchor-to-point attention decoder, including its output projection to second-order anchor displacements. By attending to all point features of its object, each anchor integrates contact information from across the surface to predict a motion update conditioned on its own position and recent displacement.

To advance the anchors from their current motion, we use the predicted second-order displacements $\widehat{\Delta^2\boldsymbol{x}}^{\,i}_{t,\mathrm{anc}}$ as learned corrections to a constant-displacement extrapolation, yielding unconstrained next positions through a Verlet-style update: $\widetilde{x}^i_{t+1,n^i_a} = 2x^i_{t,n^i_a} - x^i_{t-1,n^i_a} + \left[\widehat{\Delta^2\boldsymbol{x}}^{\,i}_{t,\mathrm{anc}}\right]_a$ Here, $[\cdot]_a$ extracts the a-th row, which contains the predicted second-order displacement of the anchor indexed by n^i_a. To recover a rigid transformation for the object, we align the current anchor positions with their predicted positions using the Kabsch algorithm (Kabsch, 1976) to obtain the recoverd rotation and translation $(\widehat{R}^i_t, \widehat{b}^i_t)$. We apply them to every surface point and the center of mass, while updating surface normals using only the rotation: $\widehat{x}^i_{t+1,n} = \widehat{R}^i_t x^i_{t,n} + \widehat{b}^i_t$, $\widehat{\nu}^i_{t+1,n} = \widehat{R}^i_t \nu^i_{t,n}$, $\widehat{c}^i_{t+1} = \widehat{R}^i_t c^i_t + \widehat{b}^i_t$.

Anchor-Supervised Motion Training. Applying the recovered rigid transformation to the current anchors yields $\widehat{x}^i_{t+1,n^i_a}$. For the fixed anchor indices $\{n^i_a\}_{a=1}^A$, we compute the ground-truth and

[2] The weights are normalized over valid candidates only, while points with no valid candidate receive a zero contact embedding. The fused representation is then normalized using RMSNorm, yielding $h^{i,0}_{t,n}$ for subsequent intra-object reasoning.

rigid-projected second-order displacements, respectively, as:

$$\left[\Delta^2 \boldsymbol{x}^{i,*}_{t,\text{anc}}\right]_a = x^{i,*}_{t+1,n^i_a} - 2x^{i,*}_{t,n^i_a} + x^{i,*}_{t-1,n^i_a}, \qquad \left[\widehat{\Delta^2 \boldsymbol{x}}^{\,i,\text{rigid}}_{t,\text{anc}}\right]_a = \widehat{x}^{i}_{t+1,n^i_a} - 2x^{i,*}_{t,n^i_a} + x^{i,*}_{t-1,n^i_a}. \quad (5)$$

Here, $*$ denotes ground-truth quantities and $[\cdot]_a$ extracts the a-th row. We supervise both the direct prediction and its rigid-projected counterpart in the following objective:

$$\mathcal{L} = \ell_{\text{SL1}}\left(\widehat{\Delta^2 \boldsymbol{x}}^{\,i}_{t,\text{anc}}, \Delta^2 \boldsymbol{x}^{i,*}_{t,\text{anc}}\right) + \ell_{\text{SL1}}\left(\widehat{\Delta^2 \boldsymbol{x}}^{\,i,\text{rigid}}_{t,\text{anc}}, \Delta^2 \boldsymbol{x}^{i,*}_{t,\text{anc}}\right). \quad (6)$$

where ℓ_{SL1} denotes the smooth-L1 loss. Both loss terms are computed in the normalized target space. Further implementation details are provided in Appendix B.1.

4 Experiment

Datasets & Metrics. We evaluate on MOVi-Sphere, MOVi-A, and MOVi-B (Greff et al., 2022). The datasets span increasing geometric diversity, from spheres to primitive shapes and more complex object geometries. We report autoregressive position RMSE (m) and orientation RMSE (degrees) at 50, 75, and 100 frames (Allen et al., 2023; Rubanova et al., 2024; Wei & Fink, 2025). We additionally measure excess penetration frequency and mean penetration depth relative to the ground-truth trajectories in each dataset. More details of metrics can be found in appendix C.3. The best results are highlighted in bold, while the second-best results are indicated by underlined italics.

Baselines. We compare RiCo with various baselines: mesh-based simulators (e.g. FIGNet, HOPNet), point-based methods (VPD (Whitney et al., 2024)), transformer-based approaches (HCMT (Yu et al., 2024)) and point-based simulator (RigidFormer (Dou et al., 2026)). For the main benchmark and cross-dataset generalization experiments, we adopt the results reported by RigidFormer, including its reimplementations of FIGNet, HCMT, and VPD. For the remaining experiments, we focus on HOPNet (Wei & Fink, 2025) and RigidFormer, two recent methods that are most relevant baselines of mesh-based and point-cloud-based methods, respectively. We use the officially released HOPNet checkpoint and our reimplementation of RigidFormer, as its official code is not published.

4.1 Motion Prediction Performance Across Rollout Horizons

We evaluate the autoregressive prediction accuracy of RiCo against various baselines across different object geometries and rollout horizons.

As shown in Table 1, RiCo consistently achieves lower position and orientation errors than all baselines on MOVi-A and MOVi-B across all evaluated horizons. On MOVi-A, compared with the strongest baseline, our model reduces position RMSE by approximately 35% and orientation RMSE by 38–43% over 50–100 prediction steps. Similarly, on MOVi-B, we obtain a 31–32% reduction in position error and a 35–38% reduction in orientation error. On the geometrically simpler MOVi-Sphere benchmark, our translational accuracy is comparable to RigidFormer, differing by at most 0.001 m at each horizon, while consistently achieving lower orientation error. These results suggest that combining local contact representations with intra-object contact reasoning is particularly effective for predicting rigid-body interactions across diverse object shapes, while maintaining competitive performance in simpler spherical scenarios.

Notably, the prediction improvements persist throughout the evaluated autoregressive rollout horizons. At 100 frames, RiCo achieves position and orientation RMSEs of 0.115 m/11.40° on MOVi-A and 0.111 m/9.54° on MOVi-B, respectively, substantially outperforming the corresponding baselines. These results highlight the effectiveness of our contact-conditioned point Transformer, which integrates sparse local contact information through intra-object contact reasoning, for long-horizon rigid-body motion prediction.

4.2 Generalization Across Object Geometries

A useful rigid-body dynamics model must predict interactions across diverse object geometries, including shapes that differ from those in its training data. We therefore evaluate how well dynamics learned from one geometry distribution transfer to another.

As shown in Table 2, RiCo exhibits an asymmetric generalization pattern, with particularly strong performance when transferring from geometrically diverse training distributions to simpler ones. When trained on the geometrically diverse MOVi-B dataset, our model transfers particularly well to its simpler constituent distributions. From MOVi-B to MOVi-Sphere, our 100-step position RMSE is

Table 1: **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.

Model	MOVi-A			MOVi-B			MOVi-Sphere		
	50	75	100	50	75	100	50	75	100
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$_{reimpl}$	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	*0.049*/*5.06*	*0.103*/*10.90*	*0.177*/*18.32*	0.050/*3.97*	*0.095*/*8.51*	*0.161*/*15.33*	**0.026**/*3.00*	**0.057**/*6.48*	**0.099**/*11.19*
RiCo	**0.031/2.86**	**0.067/6.49**	**0.115/11.40**	**0.032/2.59**	**0.065/5.45**	**0.111/9.54**	**0.026/2.65**	*0.058*/**5.59**	*0.100*/**9.69**

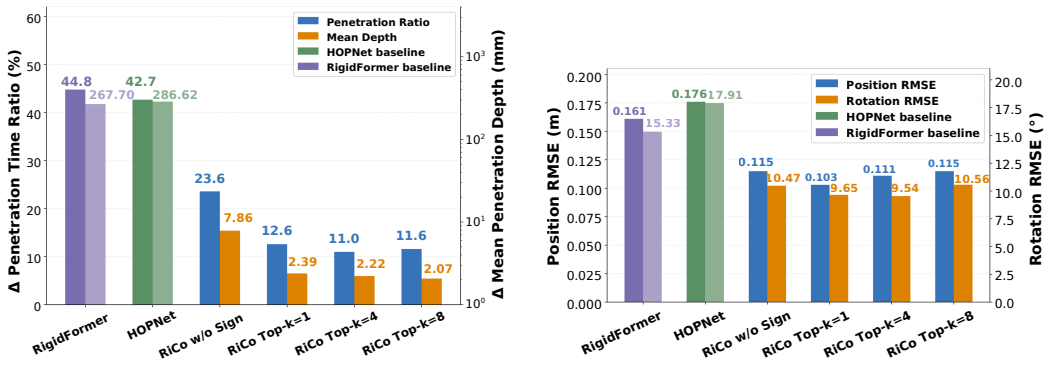
Note: The subscript *reimpl* denotes reimplementations conducted by Rigidformer authors.

0.085 m, compared with 0.160 m for HopNet and 0.207 m for RigidFormer. Similarly, from MOVi-B to MOVi-A, our error is 0.112 m/11.26°, substantially lower than both baselines. Notably, the additional geometry diversity in MOVi-B does not lead to negative transfer on simpler shapes. We observe similar improvements when transferring from MOVi-A to MOVi-Sphere. In contrast, generalization from geometrically restricted training distributions to more diverse target distributions remains challenging. When transferring from MOVi-A to MOVi-B, RiCo achieves the lowest position error, although RigidFormer retains a slightly lower orientation error. When trained exclusively on MOVi-Sphere, our method does not consistently outperform RigidFormer on the more geometrically diverse MOVi-A and MOVi-B benchmarks.

These results suggest that combining local contact representations with intra-object contact reasoning supports effective transfer across different geometry distributions, particularly when the training data provides sufficiently diverse geometric coverage. Nevertheless, generalization to more complex geometries remains dependent on the diversity of the training distribution.

Table 2: **Generalization performance across MOVi-Sphere (S), MOVi-A (A), and MOVi-B (B).** Each reports position RMSE (m) / orientation RMSE (°) at prediction horizons of 75 and 100 frames.

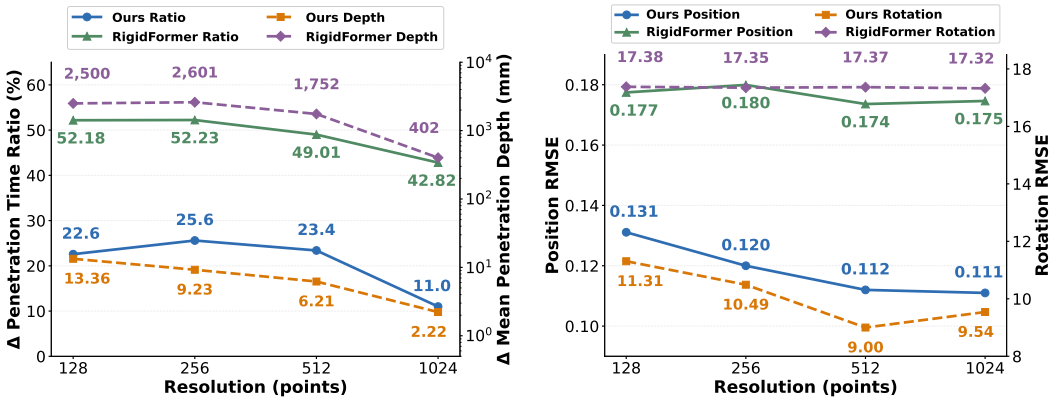
Train	Test	HopNet		RigidFormer		RiCo	
		75	100	75	100	75	100
MOVi-S	**MOVi-A**	0.112/11.29	0.197/*17.74*	**0.107**/*11.27*	**0.183**/17.92	*0.110*/**10.15**	*0.190*/**16.79**
	MOVi-B	0.106/9.75	0.188/17.13	**0.096/9.42**	**0.161**/*16.81*	*0.103*/*9.63*	*0.170*/**16.52**
MOVi-A	**MOVi-S**	0.100/9.16	0.172/15.17	*0.087*/*7.94*	*0.153*/*14.27*	**0.055/5.63**	**0.094/9.83**
	MOVi-B	*0.117*/10.77	*0.202*/18.66	0.123/**9.29**	0.208/**17.08**	**0.095**/*10.56*	**0.164**/*17.93*
MOVi-B	**MOVi-S**	*0.095*/*9.09*	*0.160*/*15.04*	0.123/10.52	0.207/17.88	**0.049/5.31**	**0.085/9.29**
	MOVi-A	*0.120*/*12.08*	*0.202*/*19.15*	0.140/12.25	0.234/20.28	**0.066/6.38**	**0.112/11.26**



(a) Relative penetration statistics.

(b) MOVi-B prediction error.

Figure 3: **Ablation of local contact reasoning on MOVi-B.** (a) Ground-truth-relative penetration statistics. (b) Position and orientation RMSE at the 100-step prediction horizon. For each method, the left bar shows the time ratio (left y-axis), and the right bar shows penetration depth (right y-axis).



(a) Relative penetration statistic.

(b) MOVi-B prediction error.

Figure 4: **Resolution and efficiency analysis on MOVi-B.** (a) Ground-truth-relative penetration statistics across different point-cloud resolutions. (b) Position and orientation prediction errors. For these curves, solid lines denote time ratios, measured on the left y-axis. Dashed lines denote penetration depths, measured on the right y-axis.

4.3 In-Depth Study on the Contact Fidelity

While trajectory RMSE measures prediction accuracy, it does not directly measure the physical consistency of predicted interactions. To evaluate contact fidelity, we introduce ground-truth-relative penetration metrics. It accounts for the geometric overlaps that already present in ground-truth trajectories. Specifically, we report the differences in penetration-time ratio (ΔPTR) and mean penetration depth (ΔMPD) between predicted and ground-truth trajectories. Detailed implementation can be seen in appendix C.3. For these analyses, we evaluate RiCo against the released HOPNet checkpoint and our independent RigidFormer reimplementation using the same penetration evaluation procedure. We further investigate the effects of signed geometric information, contact neighborhood size, and point-cloud resolution on prediction accuracy and contact fidelity. We also conduct runtime analysis on point-cloud resolution in appendix C.4.

Contact fidelity. As shown in Fig. 3a, RiCo achieves substantially lower ground-truth-relative penetration statistics than both baselines. On MOVi-B, RiCo records a ΔPTR of 11.0% and a ΔMPD of 2.22 mm, compared with 42.7% and 286.62 mm for HOPNet, and 44.8% and 267.70 mm for our RigidFormer reimplementation. Together with the lower trajectory errors, these results indicate that RiCo improves prediction accuracy, while maintains high contact fidelity.

Ablation of local contact representation. As detailed in Sec. 3.1, our local contact representation incorporates multiple geometric, motion, and physical features, most of which are either standard physical properties or have been explored in prior works, such as relative geometry and motion (Wei & Fink, 2025; Dou et al., 2026). Therefore, we focus on evaluating the contribution of our estimated signed distance and investigating the effect of contact neighborhood size.

Estimated signed distance. To investigate the importance of signed geometric information, we replace the estimated signed distance with its absolute value while retaining all other input features. As shown in Fig. 3, this ablation only moderately affects trajectory RMSE but substantially degrades contact fidelity. Specifically, ΔPTR increases from 11.0% to 23.6%, while ΔMPD increases from 2.22 to 7.86 mm. These results suggest that explicitly encoding the signed geometric relationship between nearby surfaces helps the model better capture contact configurations and reduce interpenetration, even when its overall trajectory prediction accuracy changes only marginally.

Number of contact neighbors. We further vary the number of retained contact candidates $K \in \{1, 4, 8\}$. As shown in Fig. 3, the model maintains relatively stable prediction accuracy and penetration statistics across different neighborhood sizes. While $K = 8$ achieves slightly lower ΔMPD, $K = 4$ achieves the lowest ΔPTR and orientation RMSE. These results suggest that a compact contact neighborhood is sufficient to achieve strong prediction performance under our benchmark.

Exploiting high-resolution contact geometry. We next investigate whether increasing point-cloud resolution provides additional benefits for contact modeling, as shown in Fig. 4a and 4b. Increasing

RiCo's resolution from 512 to 1024 points yields only a marginal change in position RMSE, from 0.112 to 0.111 m, but substantially improves contact fidelity. Specifically, ΔPTR decreases from 23.4% to 11.0%, while ΔMPD decreases from 6.21 to 2.22 mm.

In comparison, our RigidFormer reimplementation exhibits nearly unchanged trajectory RMSE across different resolutions, while its penetration statistics remain considerably higher despite improvements from increased surface sampling. These suggest that RiCo's local contact reasoning architecture is more effective in exploiting the richer contact information provided by high-resolution geometry. Compared with RigidFormer's shape compression and global interaction reasoning, RiCo retains fine-grained surface information through sparse local contact representations, supporting more accurate contact modeling and lower trajectory errors.

4.4 Zero-shot Scalability to Large-scale Scenes

We further investigate whether the learned dynamics can zero-shot generalize to large-scale scenes containing more interacting objects than those encountered during training. Inspired by the large-scale evaluation setting of SDF-Sim (Rubanova et al., 2024), we construct three large-scale scenarios, each containing 270 rigid objects. RiCo is trained exclusively on MOVi-B scenes containing 3–10 objects and directly evaluated on these large-scale scenarios.

Table 3: **Large-scale prediction.**

	Position / Orientation RMSE			Runtime
Scene	**120**	**240**	**360**	**FPS**
Spheres	0.029/1.73	0.243/43.07	0.713/89.90	5.02
Spots	0.019/1.21	0.211/22.90	0.785/86.74	6.50
Knots	0.017/0.81	0.139/13.43	0.843/78.00	6.54

As shown in Table 3, we evaluate the autoregressive prediction accuracy at 120, 240 and 360 steps. Since collision events occur mainly during the latter part of these trajectories, we extend the evaluation horizon to 360 steps to further examine long-term prediction accuracy after extensive object interactions. More results can be found from appendix C.5. Compared with RigidFormer, which trains on large-scale WreckingBall scenes containing up to 217 objects, our evaluation directly applies a model trained on small-scale scenes to substantially larger configurations. Although RigidFormer reports a throughput of approximately 20 FPS on its large-scale benchmark, its experimental setting differs in object geometry and training settings.

4.5 Transfer from Simulated to Real-World Dynamics

To further investigate whether the learned local contact dynamics can generalize beyond simulated environments, we evaluate RiCo on real-world multi-ball collision experiments. In Figure 8, our real-world experiments are conducted on a billiard table with an effective playing area of 2.54×1.27 m, using balls with a radius of 33.5 mm. We construct a corresponding simulated environment with comparable geometric configurations and train RiCo exclusively on simulated trajectories. The trained model is evaluated zero-shot on five real-world experimental sessions. Further details and qualitative results of the experimental setup and simulation environment are provided in the appendix C.6. Quantitatively, RiCo achieves position RMSEs of 138.54, 214.69, and 272.42 mm at prediction horizons of 50, 75, and 100 frames. For reference, the physics engine achieves corresponding RMSEs of 65.37, 147.84, and 182.96 mm. We observe discrepancies when reconstructing real-world ball configurations in simulation using camera-estimated positions, suggesting that camera-based localization inaccuracy is a major source of error in our sim-to-real evaluation. These results provide preliminary evidence that the contact dynamics learned through sparse local geometric reasoning can transfer from simulation to real-world multi-object interactions.

5 Conclusion

In this work, we introduce RiCo, a neural simulator that models rigid-body interactions through sparse local contact reasoning. By preserving fine-grained contact geometry while avoiding dense scene-level interactions, our approach achieves accurate and physically consistent predictions with high computational efficiency. Experiments demonstrate its ability to exploit high-resolution geometry, generalize across diverse object distributions, scale to scenes containing 270 objects and transfer to real world. Nevertheless, our current framework focuses on passive rigid-body dynamics and does not account for deformable objects or actuated systems. Extending local contact reasoning to these more complex physical interactions remains an important direction for future work.

AI USE STATEMENT

Generative AI tools were used to assist with code development, debugging, experimental implementation, result analysis, manuscript preparation, and figure design. The authors directed the research, developed the core methodological ideas, designed the experiments, and critically evaluated the AI-assisted outputs. All reported experimental results and scientific claims were reviewed and verified by the authors. The authors take full responsibility for the methodology, implementation, experimental results, and final manuscript.

ETHICS STATEMENT

This work investigates neural simulation of multi-object rigid-body dynamics using synthetic datasets and real-world billiard experiments. The study does not involve human subjects, personal data, or privacy-sensitive information. While neural simulation may have broader applications in robotics and autonomous systems, our work focuses on fundamental research in physical dynamics modeling. We are not aware of any immediate ethical concerns arising from the methodology or experimental settings presented in this paper.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of our model architecture and local contact reasoning framework in the main paper, with additional implementation details in the appendix. The experimental setup, evaluation metrics, baseline configurations, and ablation studies are documented in the experimental sections and supplementary materials. We additionally describe the dataset configurations, training procedures, and evaluation protocols, including the zero-shot real-world experiments, in the appendix to facilitate reproduction of our results.

REFERENCES

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Kelsey R. Allen, Yulia Rubanova, Tatiana Lopez-Guevara, William Whitney, Alvaro Sanchez-Gonzalez, Peter W. Battaglia, and Tobias Pfaff. Learning rigid dynamics with face interaction graph networks. In *International Conference on Learning Representations, ICLR*, 2023.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629. IEEE, 2023.

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.

Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *The Tenth International Conference on Learning Representations, ICLR*, 2022.

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal M. P. Behbahani, Stephanie C. Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott E. Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *International Conference on Machine Learning, ICML*, volume 235, pp. 4603–4623, 2024.

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.

Zhiyang Dou, Minghao Guo, Haixu Wu, Doug Roble, Tuur Stuyck, and Wojciech Matusik. Rigidformer: Learning rigid dynamics using transformers. *arXiv preprint arXiv:2605.09196*, 2026.

Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3749–3761, 2022.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, volume 97, pp. 2555–2565, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20765–20779, June 2026.

Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Foundations of Crystallography*, 32(5):922–923, 1976.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018.

Sixu Lin, Junliang Chen, Huaiyuan Xu, Zhuohao Li, Guangming Wang, Yixiong Jing, Sheng Xu, Runyi Zhao, Brian Sheil, Lap-Pui Chau, et al. Roboflow4d: A lightweight flow world model toward real-time flow-guided robotic manipulation. *arXiv preprint arXiv:2605.17522*, 2026.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

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*, 2017.

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*, 2024.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *International conference on machine learning (ICML)*, 2018.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning (ICML)*, 2020.

Amaury Wei and Olga Fink. Integrating physics and topology in neural networks for learning rigid body dynamics. *Nature Communications*, 16(1):6867, Jul 2025. ISSN 2041-1723. doi: 10.1038/s41467-025-62250-7. URL https://doi.org/10.1038/s41467-025-62250-7.

William Whitney, Tatiana Lopez-Guevara, Tobias Pfaff, Yulia Rubanova, Thomas Kipf, Kimberly Stachenfeld, and Kelsey Allen. Learning 3d particle-based simulators from rgb-d videos. In *International Conference on Learning Representations (ICLR)*, 2024.

Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. Slotformer: Unsupervised visual dynamics simulation with object-centric models. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Youn-Yeol Yu, Jeongwhan Choi, Woojin Cho, Kookjin Lee, Nayong Kim, Kiseok Chang, ChangSeung Woo, Ilho Kim, SeokWoo Lee, Joon Young Yang, et al. Learning flexible body collision dynamics with hierarchical contact mesh transformer. In *International Conference on Learning Representations (ICLR)*, 2024.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

A Problem Formulation

Physical world modeling. A physical world model aims to capture how the physical environment evolves over time, enabling embodied agents to anticipate future states and reason about the consequences of their actions through predictive rollouts. Such predictive dynamics provide an important foundation for long-horizon reasoning, planning, and control, where an agent must evaluate possible future outcomes before acting. In physical environments, however, meaningful prediction requires not only modeling the motion of individual objects, but also reasoning about the interactions among them (Sanchez-Gonzalez et al., 2020). This is particularly important in multi-object rigid-body systems, where contact and collision govern the transfer of momentum and can induce abrupt changes in both translational and rotational motion. Accurately modeling collision dynamics is therefore a key capability for physical world models to move beyond simple motion extrapolation and faithfully predict the long-horizon evolution of interacting scenes.

Rigid Body Contact Modeling. In this work, we study the dynamics of contact between rigid bodies. We consider a scene containing M rigid objects. At time t, object i is represented by N tracked surface samples, with positions $\boldsymbol{x}_t^i \in \mathbb{R}^{N\times 3}$ and surface normals $\boldsymbol{\nu}_t^i \in \mathbb{R}^{N\times 3}$. We denote its center of mass by $c_t^i \in \mathbb{R}^3$ and its physical properties by $\phi^i = [(m^i)^{-1}, \mu^i, e^i]$, where m^i, μ^i, and e^i are the mass, friction coefficient, and coefficient of restitution, respectively.

We collect the time-varying geometric quantities into the scene state $\mathbf{s}_t$ and the time-invariant physical properties into $\boldsymbol{\phi}$. In this way, $\mathbf{s}_t = \left\{\boldsymbol{x}_t^i, \boldsymbol{\nu}_t^i, c_t^i\right\}_{i=1}^M$, and $\boldsymbol{\phi} = \left\{\phi^i\right\}_{i=1}^M$.

Surface-point identities are preserved over time, allowing us to compute their finite-difference displacements as $\Delta\boldsymbol{x}_t^i = \boldsymbol{x}_t^i - \boldsymbol{x}_{t-1}^i$. We additionally retain the reference geometry $\boldsymbol{x}_0 = \{\boldsymbol{x}_0^i\}_{i=1}^M$. The dynamics model operates directly on these surface samples without requiring mesh connectivity.

Let ξ denote the static environment together with its geometric and physical properties. Given two consecutive scene states, we learn a transition model

$$\widehat{\mathbf{s}}_{t+1} = f_\theta(\mathbf{s}_t, \mathbf{s}_{t-1}, \boldsymbol{\phi}, \xi)\,, \tag{7}$$

which predicts the next rigid-body configuration. At inference time, the same transition model is applied recursively to generate long-horizon rollouts.

B Additional Method Details

We provide implementation details of the surface-state representation, sparse contact construction, network architecture, rigid-motion projection, and training/inference procedures. Unless otherwise stated, the configurations below describe our standard fixed-resolution local-contact implementation, rather than separate resolution-ablation variants.

B.1 Surface Representation and Feature Preprocessing

Persistent surface representation. The offline dynamics model operates on object-aligned surface point sets stored in trajectory data. Its default configuration represents each active rigid body with $N = 1024$ ordered surface points and corresponding surface normals.

Point indices retain their physical correspondence across consecutive frames; therefore, a point's finite-difference displacement is obtained directly by subtracting its position at the preceding frame, rather than by performing nearest-neighbor matching across frames.

The standard MOVi training data uses $O_{\text{data}} = 10$ object slots per scene. An object-level activity mask identifies occupied slots, and inactive slots are excluded from network evaluation and losses. This fixed storage dimension is specific to the training data and does not constrain the number of objects supported by the model at inference time.

The default configuration does not use a variable-length point mask within an active object: each occupied slot contains precisely 1024 point entries. The neural runtime consumes these recorded point sets without resampling them at each prediction step.

Object state and physical parameters. For a source point $\mathbf{x}_{t,n}^i$ on object i, we use the physically defined center of mass $\mathbf{c}_t^i$ from the trajectory state, not the arithmetic mean of the sampled surface points. The finite-difference displacement is $\Delta\mathbf{x}_{t,n}^i = \mathbf{x}_{t,n}^i - \mathbf{x}_{t-1,n}^i$; it is *not* divided by the simulation time step. Surface normals are normalized to unit length before constructing the descriptors. Object-level physical parameters $\boldsymbol{\phi}^i = [(m^i)^{-1}, \mu^i, e^i]$ are broadcast to the points of that object. We further preserve the initial surface configuration $\mathbf{X}_0^i$ as a reference throughout the trajectory, so that each point also encodes its displacement from its initial position.

Table 4 specifies the exact order of the two feature vectors. The source-point descriptor has 15 channels, while each external-contact descriptor has 14 channels. The distance used for neighbor weighting is maintained separately and is not duplicated as an additional learned feature channel.

Table 4: Input feature specification. All geometric positions and finite-difference displacements are expressed in the same scene coordinate system. Here $\mathbf{y}$ is an external candidate point, $\mathbf{n}_{\text{ext}}$ its unit normal, and $\Delta\mathbf{y}$ its finite-difference displacement.

Descriptor	Channels (in implementation order)	Dim.
Source-point descriptor $\mathbf{s}_{t,n}^i$		
	$\mathbf{x}_{t,n}^i - \mathbf{c}_t^i$	3
	$\Delta\mathbf{x}_{t,n}^i$	3
	$\boldsymbol{\nu}_{t,n}^i$	3
	$(m^i)^{-1}$, μ^i, e^i	3
	$\mathbf{x}_{t,n}^i - \mathbf{x}_{0,n}^i$	3
External-contact descriptor $\boldsymbol{\eta}_{t,n,k}^i$		
	$(\mathbf{y} - \mathbf{x}) / (\|\mathbf{y} - \mathbf{x}\|_2 + \epsilon)$	3
	$(\mathbf{x} - \mathbf{y})^\top \mathbf{n}_{\text{ext}}$	1
	$\mathbf{n}_{\text{ext}}$	3
	$\Delta\mathbf{y} - \Delta\mathbf{x}$	3
	m_{ext}^{-1}, μ_{ext}, e_{ext}	3
	$\sigma_{\text{static}} \in \{0, 1\}$	1

Feature and target normalization. The implementation uses three separate channel-wise normalizers: one for the 15-dimensional source features, one for the 14-dimensional contact features, and one for the 3-dimensional second-order anchor-displacement target. Each has the form

$$\mathcal{N}(\mathbf{z}) = \frac{\mathbf{z} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}, \qquad \mathcal{N}^{-1}(\widehat{\mathbf{z}}) = \widehat{\mathbf{z}} \odot \boldsymbol{\sigma} + \boldsymbol{\mu}, \tag{8}$$

where the division and multiplication act channel-wise. Normalization statistics are fixed during model training and saved with the checkpoint; they are not recomputed from validation or test trajectories. The provided statistics-generation script accumulates moments over active objects in the

training split and computes contact statistics over valid contact entries only. It uses a shared scale for each selected three-dimensional vector group. In its physical-semantics policy, unit directions and normals, friction/restitution coefficients, and the static indicator remain in their original ranges, while the signed tangent-plane offset is divided by the contact radius. Other continuous channels and the anchor target use training-set statistics. An explicitly populated normalization configuration takes precedence over a generated dataset sidecar; consequently, the exact statistics used for a reported experiment are those saved in its checkpoint.

B.2 Sparse Contact Neighborhood Construction

Broad-phase object filtering. A contact neighborhood is recomputed from the current geometry at inference time. For each active object i, we construct an axis-aligned bounding box from its current surface points, with lower and upper corners $\mathbf{l}_i$ and $\mathbf{u}_i$. For every pair of distinct active objects (i, j), their minimum box separation is

$$d_{\mathrm{AABB}}(i,j) = \left\| \max(\mathbf{l}_i - \mathbf{u}_j, \mathbf{l}_j - \mathbf{u}_i, \mathbf{0}) \right\|_2 , \tag{9}$$

with the maximum taken element-wise. Only pairs satisfying $d_{\mathrm{AABB}}(i,j) \le \rho$ are retained, where ρ is the contact radius. By default, we adopt $\rho = 0.1$ m. The pairwise geometric search is performed without gradient tracking.

Point-level search and global candidate selection. For each retained unordered pair (i, j), the implementation computes a pairwise Euclidean-distance matrix using the surface samples of the two objects. In both directions, it retains at most K nearest external samples for every source point. The candidates obtained from all eligible neighboring objects are then merged, and the closest K dynamic candidates are retained for each source point. Searches over eligible object pairs are processed in bounded batches (`pair_batch_size`$= 16$ in the default configuration)

The default MOVi setting also includes a static horizontal plane at height z_{table}. For a source point $\mathbf{x} = (x, y, z)$, its static candidate is the analytic projection $\mathbf{y}_{\mathrm{table}} = (x, y, z_{\mathrm{table}})$, with normal $(0, 0, 1)$, zero displacement, zero inverse mass, and the plane's assigned friction and restitution. This analytic candidate is combined with the K retained dynamic candidates, and a final nearest-K selection is performed across both types. A selected candidate is valid only if it comes from an eligible object or the plane and has a current Euclidean distance no greater than ρ. Contacts with the source object's own points are excluded. The default number of retained contact candidates is $K = 4$.

Estimated signed offset and contact masking. For a selected candidate $\mathbf{y}$ with unit outward normal $\mathbf{n}_{\mathrm{ext}}$, we compute

$$\delta = (\mathbf{x} - \mathbf{y})^\top \mathbf{n}_{\mathrm{ext}}. \tag{10}$$

This quantity is the signed offset from the *candidate's local tangent plane*; it is an estimated local contact descriptor, not an exact continuous-mesh signed-distance function. Its sign depends on the orientation of the supplied surface normal. Together with the unit direction, relative finite-difference displacement, external material parameters, and static flag, it forms the 14-dimensional descriptor specified in Table 4. The feature builder uses $\epsilon = 10^{-8}$ in the direction normalization.

Offline contact caches versus online contact queries. The training dataset stores a final Top-K target-index tensor at each recorded frame. During teacher-forced training, the implementation reads these target indices and reconstructs the relative contact features using the recorded current point positions, displacements, normals, and physical parameters; it does not redo nearest-neighbor search in every training iteration. Invalid cached candidates are encoded with a sentinel index and masked. By contrast, autoregressive inference invokes the live broad-phase and nearest-neighbor query on the model's newly predicted geometry at every step, allowing neighborhoods to change as contacts appear and disappear. The contact-index selection itself is non-differentiable; the trainable contact encoder and subsequent network operate on the selected descriptors.

B.3 Network Architecture

Pointwise encoders and distance-weighted fusion. The source-point encoder is a two-layer MLP $15 \to 128 \to 384$ with a GELU activation. Each 14-dimensional contact descriptor is independently

embedded by another two-layer MLP $14 \to 128 \to 384$, using a Softplus activation. The contact embeddings are aggregated by a masked, distance-dependent softmax. In the implementation, the learnable temperature is parameterized as

$$\tau = \mathrm{softplus}(\tau_{\mathrm{raw}}) + \tau_{\min}, \qquad \tau_{\min} = 10^{-4}, \tag{11}$$

and initialized so that $\tau = 0.005$. Softmax probabilities are computed over valid candidates only and renormalized after masking; a source point without any valid candidate receives a zero contact embedding. The sum of the source and aggregated contact embeddings is then normalized with RMSNorm to form the input point token.

Shared intra-object Transformer. Each active object's N tokens are processed independently by the same six-block point Transformer. Every block applies pre-normalized, 12-head self-attention followed by a separate pre-normalized feed-forward network, with residual connections around both components. At token width $d = 384$, each attention head has dimension 32. For the gated-attention configuration, the implementation first computes standard scaled dot-product attention and then multiplies the attended features by a learned sigmoid gate computed from the normalized input tokens, before the attention output projection. The feed-forward network uses a $384 \to 3072$ projection followed by GLU, producing a hidden width of 1536, and a final $1536 \to 384$ projection. The attention and feed-forward dropout settings are both zero in the default configuration. No attention operation in this encoder directly connects point tokens belonging to different objects.

Anchor-query decoder. The decoder creates one query for each of $A = 8$ persistent anchor indices. A query combines (i) the encoded token at that anchor and (ii) a geometry embedding of the anchor's six-dimensional state, consisting of its position relative to the physical center of mass and its preceding finite-difference displacement. The geometry encoder is an MLP $6 \to 128 \to 384$ with GELU. The resulting anchor queries pass through six cross-attention blocks with 12 heads, attending to all N tokens of the *same* object. Like the point Transformer, each cross-attention block uses separate query/token RMSNorm, gated attention, residual connections, and a GLU feed-forward network of hidden width 1536. An RMSNorm and an output MLP $384 \to 128 \to 3$ with GELU produce the predicted second-order displacement of every anchor. Decoder parameters are shared across all objects.

Table 5: Default local-contact model architecture. The two Transformer stacks operate within each object; the indicated cross-attention is from anchor queries to the object's own point tokens.

Component	Configuration
Maximum object slots / points per object	10 / 1024
Anchors / contact candidates per point	8 / 4
Source feature MLP	$15 \to 128 \to 384$, GELU
Contact feature MLP	$14 \to 128 \to 384$, Softplus
Contact pooling	Masked distance softmax
Contact temperature (initial/minimum)	0.005 / 10^{-4}
Feature fusion	Sum + RMSNorm
Point Transformer	6 blocks, 12 heads
Anchor cross-attention decoder	6 blocks, 12 heads
Token / per-head dimension	384 / 32
Transformer feed-forward hidden width	1536, GLU
Attention and FFN dropout	0
Anchor geometry MLP	$6 \to 128 \to 384$, GELU
Output head	RMSNorm, $384 \to 128 \to 3$
Kabsch numerical threshold	10^{-6}

B.4 Anchor-Based Rigid Motion Prediction

Persistent anchor indices and displacement decoding. The trajectory interface provides eight fixed, distinct anchor indices for each active object. These same indices are used when constructing the anchor queries, supervising predictions, and aligning the predicted configuration; the network does not dynamically reselect anchors in the middle of a rollout. Let $\mathbf{q}_{t,a}^{i}$ be the current position of

anchor a on object i. The decoder outputs a three-dimensional *second finite difference*, denoted by $\widehat{\Delta^2 \mathbf{q}}^{\,i}_{t,a}$, without division by the physical time step squared. The unconstrained next anchor is

$$\widetilde{\mathbf{q}}^{i}_{t+1,a} = 2\mathbf{q}^{i}_{t,a} - \mathbf{q}^{i}_{t-1,a} + \widehat{\Delta^2 \mathbf{q}}^{\,i}_{t,a}. \tag{12}$$

The predicted normalized decoder output is converted back to physical units by the target normalizer before this update.

Proper rigid alignment. We recover a single proper rigid transform independently for each active object through unweighted least-squares alignment between its current anchor positions and the unconstrained next positions. With column-vector convention, let $\bar{\mathbf{q}}$ and $\bar{\widetilde{\mathbf{q}}}$ be their respective centroids, and define

$$\mathbf{H} = \sum_{a=1}^{A} (\mathbf{q}_{t,a} - \bar{\mathbf{q}})(\widetilde{\mathbf{q}}_{t+1,a} - \bar{\widetilde{\mathbf{q}}})^{\top}, \qquad \mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}. \tag{13}$$

To exclude reflections, the implementation computes

$$\mathbf{R} = \mathbf{V}\,\mathrm{diag}\big(1, 1, \mathrm{sign}\det(\mathbf{V}\mathbf{U}^{\top})\big)\,\mathbf{U}^{\top}, \qquad \mathbf{b} = \bar{\widetilde{\mathbf{q}}} - \mathbf{R}\bar{\mathbf{q}}. \tag{14}$$

The same $\mathrm{SE}(3)$ transform advances all object surface points and its physical center of mass, while normals are updated by rotation alone and renormalized to unit length. For inactive object slots, the alignment implementation returns an identity rotation and zero translation; their output states are subsequently masked. The Kabsch implementation uses a numerical threshold of $\epsilon_{\mathrm{K}} = 10^{-6}$ for masking and safe centroid computation.

Autoregressive Inference. At inference time, we initialize RiCo with two consecutive observed states at times $t-1$ and t to predict the state at $t+1$. At each step t, the recovered rotation $\widehat{R}^{i}_{t}$ and translation $\widehat{b}^{i}_{t}$ update the surface positions $\widehat{x}^{i}_{t+1,n}$ and center of mass $\widehat{c}^{i}_{t+1}$, while the rotation alone updates the surface normals $\widehat{\nu}^{i}_{t+1,n}$. We then recompute the sparse local contact representation from the predicted geometry at $t+1$ for the next transition.

C Experimental Setup and Implementation

We describe the offline data interface, optimization setup, evaluation protocol, and metric implementations used in our local-contact experiments. The architectural details of the standard model are provided in Appendix B.

C.1 Benchmarks and Data Processing

Benchmark suites. We evaluate on MOVi-Sphere, MOVi-A, and MOVi-B using the publicly released datasets from HOPNet (Wei & Fink, 2025). These benchmarks span different geometric distributions, enabling evaluations of in-distribution rollout prediction and cross-dataset generalization. Our standard offline implementation consumes precomputed trajectories rather than rendering observations or resampling surface points during training. We follow the original train/test partitions provided by HOPNet, with training and evaluation trajectories stored separately.

Trajectory representation. The standard MOVi training datasets are stored in HDF5 format, using $O_{\mathrm{data}} = 10$ object slots per scene, $N = 1024$ ordered surface points per active object, and $A = 8$ fixed anchor indices per object. This fixed object dimension is specific to the offline training data representation and does not impose a limit on the number of objects supported by RiCo.

For a trajectory with T stored frames, the primary state tensors have conceptual shapes $[T, O_{\mathrm{data}}, N, 3]$ for surface positions and normals, $[T, O_{\mathrm{data}}, 3]$ for physical centers of mass, and $[T, O_{\mathrm{data}}]$ for inverse mass, friction, restitution, and object activity. Contact-target indices are stored as $[T, O_{\mathrm{data}}, N, K]$, where $K = 4$ in the standard setting; these indices refer to neighboring surface samples or designated static contacts.

The dataset also stores trajectory lengths, terminal flags, and persistent anchor indices. The loader uses trajectory lengths to avoid padded trailing frames and excludes training windows that cross a

Table 6: Reference offline optimization settings for the local-contact model.

Setting	Value
Optimizer	AdamW
Global batch size	128
Per-device micro-batch size	8
Context / target frames	2 / 1
Precision	`bf16`
Initial / final learning rate	10^{-4} / 10^{-5}
Learning-rate schedule	Cosine decay
Weight decay	0
Gradient-norm clipping	1.0
Training epochs	60
Autoregressive fine-tuning	Disabled
Direct / projected loss weights	1.0 / 1.0
Random seed	0

terminal transition. Its two-frame input and one-frame target configuration produces three-frame training windows.

Since RiCo shares network parameters across objects, its inference is not restricted to the object count used during training. This enables our zero-shot evaluation on scenes containing 270 objects, as described in Sec. 4.4.

Cached contacts during training. For a ground-truth training window, the loader reads the cached contact target IDs corresponding to its current frame. Their positions, normals, finite-difference displacements, and physical parameters are then gathered from the current ground-truth state to reconstruct the contact descriptors. This avoids repeating the discrete nearest-neighbor search for every teacher-forced training instance. In contrast, autoregressive evaluation reconstructs contact neighborhoods from the model's *predicted* current geometry at each new step. Consequently, the model does not use future ground-truth contact assignments during inference. Additional feature and target normalization details are given in Appendix B.1.

C.2 Optimization and Training Configuration

Offline supervised training. The standard training loop receives two consecutive ground-truth states and predicts the next state with teacher forcing. The optimizer is AdamW. For each valid training object, supervision is applied to the direct and rigid-projected second-order anchor displacements using coordinate-wise smooth-$L1$ losses in normalized target space; both losses have unit weight. Objects contribute only if their slots are active at both the current and target frames. Loss numerators are summed over valid anchor coordinates.

Reference configuration. Table 6 summarizes the reference offline training configuration. Configurations associated with separate benchmark training runs and ablations are stored alongside their corresponding checkpoints.

Implementation and model selection. The trainer supports gradient accumulation through micro-batches and multi-GPU training through distributed data parallelism. Its learning-rate schedule advances per optimizer update, and its `bf16` setting applies automatic mixed precision to the neural forward computation. Normalization parameters are computed from training data or loaded from an explicitly specified configuration and are saved with the resulting checkpoint. The available validation routine evaluates autoregressive predictions at physical frame indices 50, 75, and 100. These are frame indices in the stored trajectory, not a claim that exactly that many neural updates follow the two observation frames. The checkpoint selection criterion and computational resources are specified for each final experiment.

C.3 Evaluation Metrics and Measurement Procedures

Position and orientation errors. Let $\widehat{\mathbf{c}}^i_{r,t}$ and $\mathbf{c}^{i,*}_{r,t}$ be the predicted and ground-truth physical centers of mass for object i in trajectory r at evaluation frame t. Let $\mathcal{V}_{r,t}$ be the set of objects active in both states. We first compute a per-scene position error,

$$E^{\mathrm{pos}}_{r,t} = \sqrt{\frac{1}{|\mathcal{V}_{r,t}|} \sum_{i \in \mathcal{V}_{r,t}} \left\| \widehat{\mathbf{c}}^i_{r,t} - \mathbf{c}^{i,*}_{r,t} \right\|_2^2}. \tag{15}$$

For orientation, we estimate the relative proper rotation $\Delta\mathbf{R}^i_{r,t} \in \mathrm{SO}(3)$ between predicted and ground-truth surface-point configurations using rigid alignment. Its geodesic angle in degrees is

$$\theta^i_{r,t} = \frac{180}{\pi} \arccos\left[\mathrm{clip}\left(\frac{\mathrm{tr}(\Delta\mathbf{R}^i_{r,t}) - 1}{2}, -1, 1 \right) \right]. \tag{16}$$

The per-scene orientation error is then

$$E^{\mathrm{rot}}_{r,t} = \sqrt{\frac{1}{|\mathcal{V}_{r,t}|} \sum_{i \in \mathcal{V}_{r,t}} (\theta^i_{r,t})^2}. \tag{17}$$

The reported frame-specific values are the arithmetic means of the per-scene RMSEs across evaluated trajectories. Translation is expressed in meters and orientation in degrees. This distinction matters because averaging scene-level RMSEs is not generally equivalent to pooling all objects across the dataset before taking a square root.

Collision geometry and penetration depth. Penetration is evaluated separately from the network's input-point representation. The evaluation script loads the scaled collision meshes associated with each object's shape, registers the mesh to the frame-zero ground-truth surface points, and uses rigidly estimated poses to place the mesh in every subsequent ground-truth or predicted frame. For an object pair, an AABB check first rejects pairs with disjoint boxes. For the remaining pairs, the script tests the vertices of each mesh for occupancy inside the other mesh and queries the distances of inside vertices to that mesh's surface. It takes the maximum measured depth over the two query directions. Penetration of the horizontal ground plane is evaluated from the minimum height of the transformed mesh vertices. For each active object and frame, the recorded depth $D^i_{r,t} \geq 0$ is the maximum across all detected object–object and object–ground violations; $P^i_{r,t} = \mathbb{1}[D^i_{r,t} > 0]$ indicates a detected penetration. These measurements use a geometric vertex-query approximation on the evaluation meshes, rather than an exact continuous-surface intersection-depth solver. No additional positive depth threshold is applied in the inspected script.

Ground-truth-relative penetration statistics. Let S denote either the predicted or ground-truth trajectories, and let $a^i_{r,t} \in \{0, 1\}$ indicate an active object. The evaluated frame set $\mathcal{T}$ excludes the two initial frames. Let $\mathcal{Q}$ contain all trajectory–object pairs active in at least one evaluated frame. The penetration-time ratio implemented by our evaluator is

$$\mathrm{PTR}(S) = \frac{1}{|\mathcal{Q}|} \sum_{(r,i) \in \mathcal{Q}} \frac{\sum_{t \in \mathcal{T}} a^i_{r,t} P^{i,S}_{r,t}}{\sum_{t \in \mathcal{T}} a^i_{r,t}}. \tag{18}$$

Thus, the evaluator first computes the fraction of penetrating active frames for each trajectory–object pair, then averages these fractions. The mean penetration depth is computed over all *detected penetrating* active object–frame observations,

$$\mathrm{MPD}(S) = \frac{\sum_{r,i} \sum_{t \in \mathcal{T}} a^i_{r,t} P^{i,S}_{r,t} D^{i,S}_{r,t}}{\sum_{r,i} \sum_{t \in \mathcal{T}} a^i_{r,t} P^{i,S}_{r,t}}, \tag{19}$$

with MPD set to zero if the denominator vanishes. Depths are computed in meters and converted to millimeters for display. Finally, the *ground-truth-relative* values printed by the current evaluation script are the differences of these aggregated statistics,

$$\Delta\,\mathrm{PTR} = \mathrm{PTR}(\mathrm{pred}) - \mathrm{PTR}(\mathrm{GT}), \qquad \Delta\,\mathrm{MPD} = \mathrm{MPD}(\mathrm{pred}) - \mathrm{MPD}(\mathrm{GT}). \tag{20}$$

The first quantity is reported in percentage points and the second in millimeters. These are differences of dataset-level summaries; they are *not* the frequency or conditional depth of a separately defined per-frame positive excess $\max(0, D_{r,t}^{i,\text{pred}} - D_{r,t}^{i,\text{GT}})$. This distinction is relevant when interpreting the physical-consistency results in the main paper.

C.4 Runtime analysis

Experimental setup. We evaluate computational efficiency on a single NVIDIA GeForce RTX 5090 GPU using the same 120 test trajectories for all methods. Runtime is measured with a batch size of one over 100-step autoregressive rollouts. Each rollout is initialized with two ground-truth frames, after which subsequent states are predicted autoregressively without accessing future ground-truth states.

We perform five warm-up iterations before timing and synchronize CUDA execution immediately before and after each measured rollout. Dataset loading, disk I/O, visualization, and rendering are excluded from the runtime measurements.

Computational efficiency. We examine whether retaining high-resolution geometry introduces substantial computational overhead. As shown in Fig. 5, we evaluate baselines and RiCo on the same 120 test trajectories using a single NVIDIA RTX 5090 GPU. Despite operating on 1024 surface points per object, RiCo achieves 47.89 FPS, slightly faster than our RigidFormer reimplementation (41.73 FPS) and over an order of magnitude faster than HOPNet (2.61 FPS). Reducing RiCo's resolution to 512 and 256 points further increases throughput to 87.69 and 130.87 FPS, respectively. These results demonstrate that sparse local contact reasoning enables high-resolution geometric modeling without sacrificing computational efficiency.

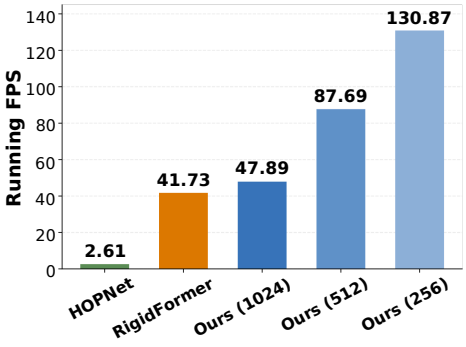


Figure 5: Runtime analysis.

Baseline implementation details. We reproduce RigidFormer for our experiments rather than using runtime measurements reported in the original paper. Although we follow its proposed architecture and inference pipeline, implementation-level differences may affect the measured throughput. Therefore, the reported RigidFormer runtime should be interpreted as the performance of our reimplementation rather than a definitive measurement of the original implementation.

For HOPNet, we introduce additional implementation optimizations to improve inference efficiency while preserving its underlying modeling approach. Consequently, the reported HOPNet throughput reflects our optimized implementation rather than its unmodified implementation.

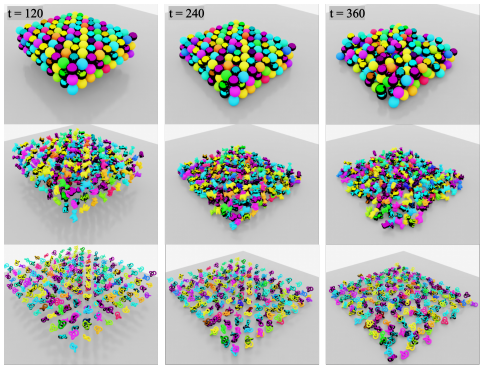
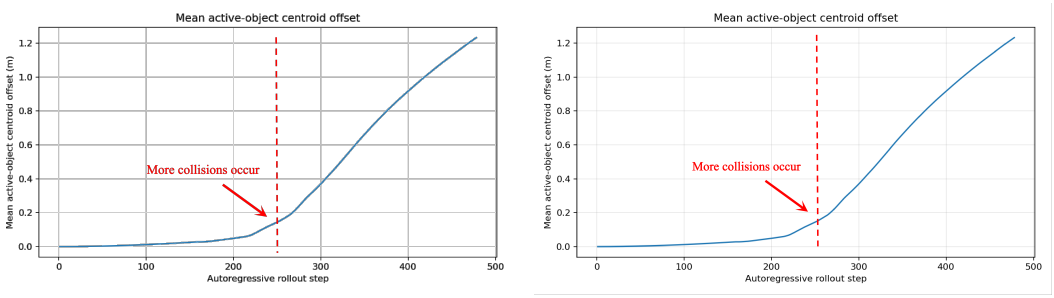


Figure 6: Qualitative results of three large-scale scenarios.

All three methods are benchmarked on the same GPU and test trajectories. Nevertheless, the reported runtime comparison should be interpreted in light of these implementation differences.

C.5 Temporal Error Accumulation in Large-Scale Scenes

To further analyze long-horizon behavior in the 270-object scenarios (Fig. 6), we record the temporal evolution of prediction errors throughout autoregressive rollouts.



(a) Position offset in the centroid of active objects. (b) Orientation Offset in the centroid of active objects.

Figure 7: **Temporal error accumulation in large-scale scenes.** Position and orientation errors throughout autoregressive rollouts of three 270-object scenarios. The dashed line indicates an approximate transition point after which object collisions become more frequent.

Figure 7 shows the position and orientation errors as functions of rollout step. In each plot, the dashed vertical line marks an approximate transition point after which collisions become more frequent, based on qualitative observation of the scene dynamics.

As shown in the figure, prediction errors remain relatively small during the earlier stage of the rollout, and then increase more rapidly after the onset of more frequent multi-object interactions. This trend is consistent across the large-scale scenarios and helps explain why we report errors not only at 120 frames, but also at longer horizons of 240 and 360 frames in Table 3.

These results complement the endpoint evaluations in the main paper by illustrating how prediction errors accumulate over time as long-horizon autoregressive simulation becomes increasingly challenging.

C.6 Real-world Experimental Setup and Simulated Training

Real-world experimental setup. We conduct real-world experiments on a standard billiard table with an effective playing area of 2.54×1.27 m. Each billiard ball has a radius of 33.5 mm. We consider multi-ball collision scenarios involving three and four balls, capturing their motion and interactions through camera-based observations.

Simulated training environment. To investigate zero-shot sim-to-real transfer, we construct a corresponding simulated billiard environment with geometric configurations comparable to those of the real-world setup. The simulation reproduces the table dimensions and ball geometry used in the experiments.

To generate diverse collision dynamics, we randomize the initial conditions of the simulated training scenarios. Specifically, the initial position of each ball is randomly perturbed within ± 0.25 m, while the initial speed of the moving ball is sampled from 1.5–2.5 m/s. Its initial moving direction is additionally perturbed within $\pm 5^\circ$ relative to the nominal direction. These randomized configurations expose the model to different contact locations, collision angles, and subsequent multi-object interactions.

RiCo is trained exclusively on 960 simulated trajectories with a duration 2s, without incorporating any real-world observations during training or performing real-world fine-tuning.

Zero-shot evaluation protocol. We evaluate the trained model on five independent real-world experimental sessions, comprising 23 valid ball samples in total. Real-world ball configurations are reconstructed using camera-estimated positions. Starting from the observed initial states, the model predicts subsequent object motion through autoregressive rollout.

We compare the predicted ball trajectories against the corresponding real-world observations and report position RMSE at evaluation frames 50, 75, and 100. For reference, we also evaluate trajectories generated by the physics engine. Both sets of predictions are evaluated against the real-world observations.

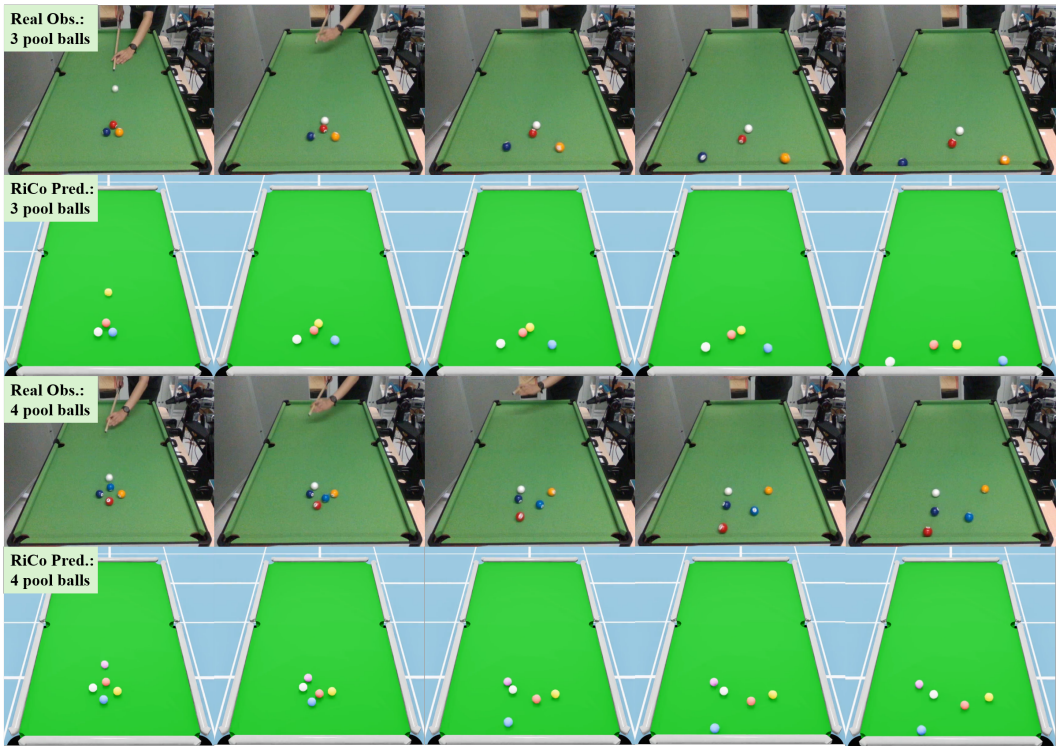


Figure 8: Illustrating 10 pairs qualitative comparison of real-world observations (upper, labeled by Real Obs.) and autoregressive predictions by RiCo (lower, labeled by RiCo Pred.) in three- and four-ball collision scenarios. RiCo's predictions are initialized by the balls' observed positions and estimated velocities in the real-world experiments, then replayed in the simulated environment. Under this setting, smaller distances between corresponding predicted and observed positions indicate greater prediction accuracy.