PIM: A LOW-LATENCY PHYSICAL INTUITION MODEL FOR PREDICTING OBJECT INTERACTION OUTCOMES

Anonymous authors
Paper under double-blind review

ABSTRACT

Predicting object interaction outcomes from initial conditions is a central problem in data-driven physical reasoning. Existing methods often use rigid-body-level states or predict the evolution through autoregressive rollouts. Coarse representations cannot represent local deformation, while rollout errors may compound over long prediction horizons. Instead, we introduce PIM, a Physical Intuition Model that directly predicts a physical system's equilibrium state from its initial mesh geometry, kinematics, and physical attributes. PIM combines local k-nearest-neighbor (KNN) attention to encode intra-object geometry, anchor-based cross-object attention to model interactions, and a dual-branch decoder that predicts object-level translations and per-vertex residual displacements. Across MuJoCo scenarios involving both rigid and deformable interactions, PIM achieves lower vertex and centroid errors than the evaluated baselines. By predicting complete equilibrium states in a single forward pass, PIM substantially reduces latency relative to autoregressive learned models and MuJoCo rollouts under the evaluated protocols, enabling low-latency inference when only the final outcome is required.

1 INTRODUCTION

Simulating how objects move, collide, and deform is important in computer graphics, robotics, and embodied intelligence. Conventional physics simulators can resolve collision, contact, friction, and deformation with high fidelity, but numerical time stepping can be computationally expensive when generating large datasets or simulating complex scenes. Learning-based models offer an alternative approach to predicting object interaction outcomes from initial conditions. Adopting the terminology of prior work on predicting block tower state (Lerer et al., 2016), we use *physical intuition* to denote the ability to infer the equilibrium state of interacting objects from their initial geometry, kinematics, and physical attributes.

Existing physical AI models use object-, particle-, graph-, or mesh-based representations primarily for stepwise dynamics prediction (Battaglia et al., 2016; Li et al., 2019; Sanchez-Gonzalez et al., 2020; Pfaff et al., 2021; Allen et al., 2023b). Instead, we present PIM, a Physical Intuition Model that takes the initial scene state as input and directly predicts the equilibrium positions of all mesh vertices. This representation accommodates both rigid-body motion and non-rigid deformation, but requires the model to jointly reason about object geometry, physical attributes, and inter-object interactions across diverse contact phenomena. Throughout this paper, *equilibrium state* denotes a stationary configuration in approximate static equilibrium, reached after transient motion has decayed and the net forces and torques are balanced up to numerical tolerance. States involving persistent motion, including dynamic equilibria, are outside the scope of this work.

In this work, PIM aims for object interactions. Each object is represented by vertex tokens augmented with object-level and scene-level conditions. Local KNN self-attention captures intra-object geometry, anchor-based cross-object attention exchanges information between objects, and a dual-branch decoder separates center displacement from vertex-level residual motion. PIM therefore predicts both global object movement and local geometric change within a unified mesh representation.

We evaluate PIM on four controlled MuJoCo scenarios spanning planar and irregular-surface collisions, structural collapse, and rigid–deformable interaction. Comparisons with direct equilibrium-state predictors and autoregressive rollout baselines, together with controlled ablations, runtime measurements, and physical-validity diagnostics, demonstrate lower equilibrium-state errors and

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.

low-latency inference under the evaluated protocols while revealing the remaining gap between geometric accuracy and valid contact configurations.

The main contributions of this work are:

- We present a unified physical intuition model that directly predicts object interaction outcomes from initial geometry, kinematic conditions, and physical attributes. By representing these outcomes as equilibrium meshes, the formulation jointly accounts for rigid-body motion and localized deformation while avoiding autoregressive rollouts when only the final interaction outcome is required.
- We introduce PIM, which takes a vertex-based representation as input and combines local KNN self-attention to encode intra-object geometry with cross-object attention over source anchors selected via farthest point sampling (FPS). The latter enables information exchange between objects through a fixed number of source tokens. A dual-branch decoder then decomposes each object's terminal vertex displacement into an object-level centroid translation and zero-mean per-vertex residuals, preventing redundant encoding of global motion while preserving the full expressiveness of the displacement field.
- We develop a composite loss that supervises terminal vertex coordinates, object centroids, and edge lengths. These loss terms guide vertex-wise motion, global placement, and local final geometry, respectively, allowing PIM to learn non-rigid deformation without imposing preservation of the initial shape.

2 Related Work

Physical intuition and learning-based dynamics prediction. Physical intuition has been studied through outcome prediction, object-centric reasoning, and benchmarks covering phenomena such as collision, rolling, sliding, and soft-body interaction (Lerer et al., 2016; Piloto et al., 2022; Bear et al., 2021). Learning-based dynamics models approximate temporal physical evolution using object-centric interaction functions and visual interaction networks (Battaglia et al., 2016; Watters et al., 2017), as well as particle- or graph-based message passing (Li et al., 2019; Sanchez-Gonzalez et al., 2020; Yang et al., 2021). Later work studies generalization, symmetry, contact handling, and rigid-body prediction (Han et al., 2022; Allen et al., 2023a; Rubanova et al., 2024), while HOPNet uses time-varying combinatorial complexes for autoregressive rigid-body prediction (Wei & Fink, 2025). Other object-centric methods learn latent representations for future-frame, trajectory, or dynamics prediction (Schmeckpeper et al., 2021; Min et al., 2021; Nakano et al., 2023; Song et al., 2025). PIM differs from these predominantly temporal formulations by predicting an explicit mesh-level equilibrium state from 3D geometry and physical conditions.

Mesh-, point-, and contact-based dynamics modeling. Mesh and point representations support learning on irregular geometric domains. Point Transformer models local point-set structure with self-attention (Zhao et al., 2021). MeshGraphNets and Boundary Graph Neural Networks learn dynamics on meshes or particle–boundary graphs (Pfaff et al., 2021; Mayr et al., 2023), while related point-sequence and mesh-surrogate methods address scene flow, point-cloud forecasting, and solid-mechanics simulation (He et al., 2022; Deshpande et al., 2024). Contact-aware approaches study differentiable simulation of cloth, deformable bodies, and fluid–rigid interactions (Newbury et al., 2024; Li et al., 2022; Huang et al., 2024; Li et al., 2023a), while learned rigid-body simulators model contact through face-based interactions (Allen et al., 2023b). PIM complements these approaches by treating mesh vertices as prediction tokens, using local KNN attention for intra-object structure and anchor-based attention for cross-object interaction.

Transformer-based physical modeling. Transformers provide a general attention-based architecture (Vaswani et al., 2017), while neural operators learn solution mappings for parametric PDEs and physical surrogate modeling (Li et al., 2021; Wang et al., 2021). Subsequent work explores Galerkin, wavelet, orthogonal, factorized, and geometry-aware attention mechanisms (Cao, 2021; Tripura & Chakraborty, 2023; Xiao et al., 2024; Li et al., 2023b; Wu et al., 2024; Alkin et al., 2024), including Transformer-based simulation on irregular meshes (Garnier et al., 2026). PIM applies this attention-based perspective to multi-object terminal prediction, combining object-specific mesh tokens with separate intra-object and inter-object interaction mechanisms.

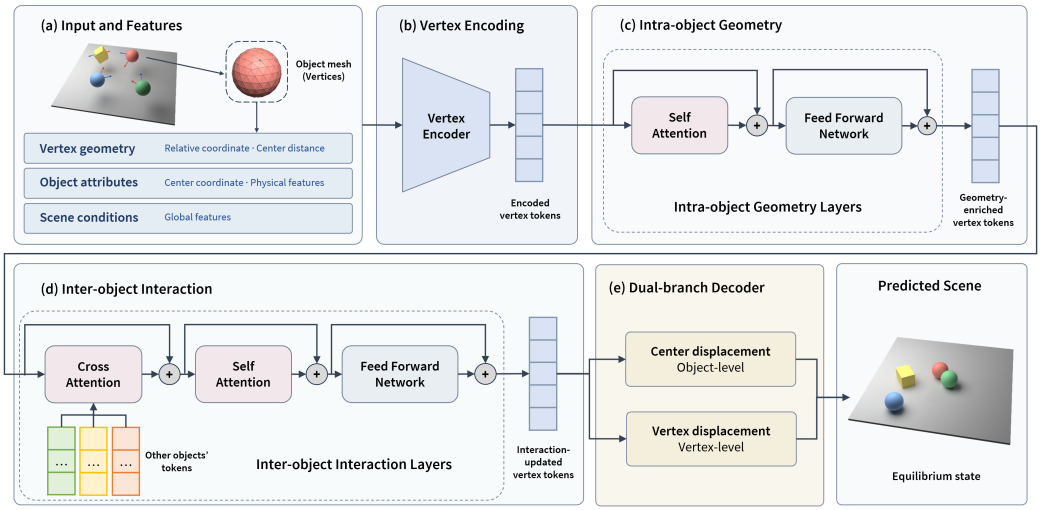


Figure 1: Overview of PIM. Initial mesh geometry, object-level physical attributes, and scene-level conditions are encoded into per-object vertex tokens. Local KNN self-attention models intra-object geometry, anchor-based cross-object attention exchanges interaction information among objects, and a dual-branch decoder predicts object-level center displacement and vertex-level residual displacement.

3 Physical Intuition Model

3.1 Problem Formulation

For a scene containing M objects, let $\mathbf{X}_i^0 \in \mathbb{R}^{n_i \times 3}$ denote the initial vertices of object i, and let $\mathbf{Y}_i$ and $\widehat{\mathbf{Y}}_i$ denote its simulated and predicted final vertices. Each object has initial velocity $\mathbf{v}_i$, an external-force vector $\mathbf{f}_i$ applied during the initial forcing interval, density ρ_i, and sliding, spinning, and rolling friction $\boldsymbol{\mu}_i = [\mu_i^t, \mu_i^s, \mu_i^r]$. We normalize force and density as $\widetilde{\mathbf{f}}_i = \mathbf{f}_i/10$ and $\widetilde{\rho}_i = \rho_i/2000$, and define the object condition $\mathbf{o}_i = [\mathbf{v}_i, \widetilde{\mathbf{f}}_i, \widetilde{\rho}_i, \boldsymbol{\mu}_i] \in \mathbb{R}^{10}$. Surface friction and normalized vertical gravity form $\mathbf{g} = [\mu_{\text{surf}}^t, \mu_{\text{surf}}^s, \mu_{\text{surf}}^r, g_z/10] \in \mathbb{R}^4$. PIM learns

$$\mathcal{F}_\theta : \left\{(\mathbf{X}_i^0, \mathbf{o}_i)\right\}_{i=1}^M \cup \{\mathbf{g}\} \longrightarrow \left\{\widehat{\mathbf{Y}}_i\right\}_{i=1}^M . \tag{1}$$

3.2 Architecture Overview

PIM first encodes initial geometry and physical conditions into per-vertex features. Intra-object layers propagate information over fixed local KNN graphs, and inter-object layers alternate anchor-based cross-object attention with local refinement. A dual-branch decoder then reconstructs the final mesh from object-level center displacement and vertex-level relative displacement (Figure 1).

3.3 Input Encoding

For object center $\mathbf{c}_i^0 = \frac{1}{n_i}\sum_p \mathbf{x}_{i,p}^0$, the encoder computes relative coordinates $\mathbf{r}_{i,p}^0 = \mathbf{x}_{i,p}^0 - \mathbf{c}_i^0$ and center distances from the object centroid, $s_{i,p}^0 = \|\mathbf{r}_{i,p}^0\|_2$. It independently projects relative coordinates, center distance, object center, object condition, and scene condition to $d/2$, $d/16$, $d/8$, $d/4$, and $d/16$ dimensions:

$$\mathbf{h}_{i,p} = \left[\phi_{\text{rel}}(\mathbf{r}_{i,p}^0), \phi_{\text{dist}}(s_{i,p}^0), \phi_{\text{cen}}(\mathbf{c}_i^0), \phi_{\text{obj}}(\mathbf{o}_i), \phi_{\text{glb}}(\mathbf{g})\right] \in \mathbb{R}^d. \tag{2}$$

A fusion projection and layer normalization yield $\mathbf{H}_{i,p}^0 = \text{LN}(\phi_{\text{fuse}}(\mathbf{h}_{i,p}))$ and the initial representation $\mathbf{H}_i^0 \in \mathbb{R}^{n_i \times d}$.

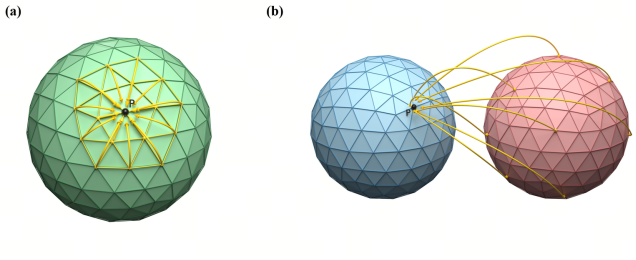


Figure 2: Attention mechanisms in PIM. (a) A target vertex aggregates from KNN neighbors in the same object. (b) A target vertex of object i aggregates from source anchors of another object j. Arrows indicate feature aggregation toward the target.

3.4 Intra- and Inter-object Attention

For intra-object geometry, a local graph is constructed once in the initial vertex space with neighborhood $\mathcal{N}_i(p)$. Each target vertex attends only to its KNN neighbors:

$$\mathbf{z}_{i,p} = \sum_{q \in \mathcal{N}_i(p)} \alpha_{p,q} \mathbf{u}_{i,q}, \qquad \alpha_{p,q} = \mathrm{softmax}_{q \in \mathcal{N}_i(p)} \left(\overline{\mathbf{q}}_{i,p}^{\mathsf{T}} \overline{\mathbf{k}}_{i,q} \right), \tag{3}$$

where queries and keys are L2-normalized, so the logits are cosine similarities without additional temperature scaling.

For cross-object interaction, each source object j supplies a anchors $\mathcal{A}_j$ selected from its initial vertices by deterministic farthest-point sampling (FPS). Sampling starts from vertex index zero and chooses the vertex farthest from the current anchor set:

$$\pi_{j,0} = 0, \qquad \pi_{j,l} = \arg\max_{q} \min_{r<l} \left\| \mathbf{x}_{j,q}^{0} - \mathbf{x}_{j,\pi_{j,r}}^{0} \right\|_2^2, \qquad l = 1, \ldots, a-1. \tag{4}$$

A target vertex of object i attends to anchors of every $j \neq i$:

$$\mathbf{z}_{i \leftarrow j,p} = \sum_{q \in \mathcal{A}_j} \beta_{p,q}^{i \leftarrow j} \mathbf{u}_{j,q}, \quad \beta_{p,q}^{i \leftarrow j} = \mathrm{softmax}_{q \in \mathcal{A}_j} \left(\overline{\mathbf{q}}_{i,p}^{\mathsf{T}} \overline{\mathbf{k}}_{j,q} \right). \tag{5}$$

Messages are averaged as $\mathbf{Z}_i^{\mathrm{cross}} = \frac{1}{M-1} \sum_{j \neq i} \mathbf{Z}_{i \leftarrow j}$, incorporated through a residual update, and followed by local attention and feed-forward refinement. Figure 2 contrasts the two attention patterns; residual block details are given in Appendix A.

3.5 Dual-Branch Decoder and Objective

The center branch mean-pools interaction-aware vertex features and predicts object translation $\Delta \mathbf{c}_i$. The relative branch predicts vertex displacements and removes their object-wise mean:

$$\Delta \mathbf{c}_i = \mathrm{MLP}_c \left(\frac{1}{n_i} \sum_{p} \mathbf{H}_{i,p}^{\mathrm{int}} \right), \qquad \widetilde{\Delta \mathbf{r}}_{i,p} = \mathrm{MLP}_r(\mathbf{H}_{i,p}^{\mathrm{int}}) - \frac{1}{n_i} \sum_{q} \mathrm{MLP}_r(\mathbf{H}_{i,q}^{\mathrm{int}}). \tag{6}$$

The final vertex is

$$\widehat{\mathbf{y}}_{i,p} = \mathbf{x}_{i,p}^{0} + \Delta \mathbf{c}_i + \widetilde{\Delta \mathbf{r}}_{i,p}. \tag{7}$$

The output layers of both decoder branches are zero-initialized, so the initial prediction equals the input mesh. For a batch of size B, coordinate and center losses are

$$\mathcal{L}_{\mathrm{coord}} = \frac{1}{3B \sum_i n_i} \sum_{b,i,p} \left\| \widehat{\mathbf{y}}_{b,i,p} - \mathbf{y}_{b,i,p} \right\|_1, \tag{8}$$

$$\mathcal{L}_{\mathrm{cen}} = \frac{1}{BM} \sum_{b,i} \left\| \widehat{\mathbf{c}}_{b,i} - \mathbf{c}_{b,i} \right\|_2^2. \tag{9}$$

We additionally supervise local terminal geometry over the fixed initial-space KNN graph, whose effective neighborhood size is $k_i = \min(k, n_i - 1)$. Let

$$\ell^{\text{pred}}_{b,i,p,q} = \|\widehat{\mathbf{y}}_{b,i,p} - \widehat{\mathbf{y}}_{b,i,q}\|_2, \qquad \ell^{\text{target}}_{b,i,p,q} = \|\mathbf{y}_{b,i,p} - \mathbf{y}_{b,i,q}\|_2.$$

The KNN edge-length loss is the mean absolute difference between predicted and target neighbor distances:

$$\mathcal{L}_{\text{edge}} = \frac{1}{B\sum_i n_i k_i} \sum_{b,i,p} \sum_{q \in \mathcal{N}_i(p)} \left| \ell^{\text{pred}}_{b,i,p,q} - \ell^{\text{target}}_{b,i,p,q} \right|. \tag{10}$$

Target edge lengths come from the final mesh, so this term applies to both rigid and deformable objects without enforcing the initial shape. The complete objective is

$$\mathcal{L} = \mathcal{L}_{\text{coord}} + \lambda_{\text{cen}}\mathcal{L}_{\text{cen}} + \lambda_{\text{edge}}\mathcal{L}_{\text{edge}}. \tag{11}$$

4 Experiments

4.1 Experimental Setup

We evaluate PIM on four MuJoCo (Todorov et al., 2012) datasets: nine-sphere interactions on a flat surface (Flat), the same interactions on irregular supporting geometry (Bumpy), impact-induced collapse of an eight-block tower (Jenga), and two rigid spheres impacting a deformable box (Deformable). A separate PIM is trained for each scenario. Dataset construction, split sizes, mesh resolutions, preprocessing, architecture settings, and training configurations are provided in Appendices B and C.

On the full datasets, we compare PIM with two direct equilibrium-state predictors developed in this study: Rigid-Pose MLP and Vertex Transformer. On a controlled Flat split with shared scenes and terminal targets, we additionally compare PIM and Small-PIM with HOPNet (Wei & Fink, 2025) and PyTorch reimplementations of FIGNet (Allen et al., 2023b; Yi, 2024) and MeshGraphNets (Pfaff et al., 2021); *-Adapt* variants additionally receive the external force applied during the initial forcing interval. We report mean vertex and object-centroid errors, physical-validity diagnostics, and Jenga generalization without fine-tuning to shifts in impactor conditions and top-block geometry. Complete baseline, metric, and generalization protocols are provided in Appendices C, D, and B.4.

4.2 Main Results

PIM obtains the lowest vertex and centroid errors among all evaluated methods in all four scenarios (Figure 4; exact values are given in Appendix Table 10). Relative to the lowest-error learned baseline for each metric and scenario, PIM reduces vertex error by 28.0%, 18.8%, 30.5%, and 40.5% on Flat, Bumpy, Jenga, and Deformable, respectively. The corresponding centroid-error reductions are 37.1%, 20.4%, 30.2%, and 31.9%. The smallest vertex-error margin occurs on Bumpy, while the largest improvement occurs on Deformable. Rigid-Pose MLP is evaluated on Deformable as a rigid-output equilibrium-state predictor, although its parameterization cannot express local non-rigid deformation. Vertex Transformer has lower edge error than PIM on Jenga and Deformable, showing that local shape preservation and terminal object placement measure different aspects of prediction quality. Figure 3 shows one prediction from each scenario, while Appendix Figures 11–14 provide additional test examples.

Jenga OOD generalization. PIM achieves the lowest vertex and centroid errors under all three initial-condition shifts (Figure 5). Relative to the lowest-error direct equilibrium-state baseline under each condition, PIM reduces vertex error by 24.6%, 18.3%, and 28.3% on position, velocity, and force OOD, respectively; the corresponding centroid-error reductions are 32.3%, 20.9%, and 29.2%. Velocity OOD is the most challenging initial-condition shift and narrows PIM's advantage most strongly. On the ID-U control, vertex and centroid errors differ from the matched-ID results by at most 3.8% and 2.5%, respectively, across the three models, indicating that the larger OOD degradation is primarily associated with the shifted initial conditions rather than the outcome-selection protocol.

We further evaluate shape generalization by replacing only top-layer block J with an equal-volume cube while retaining its vertex count, face-index topology, and the original matched interaction

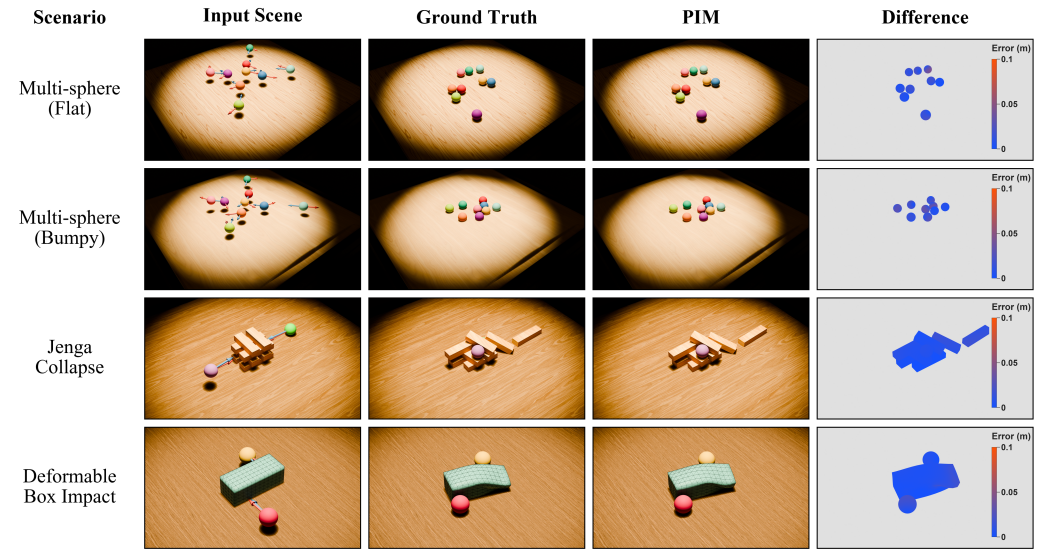


Figure 3: Qualitative final-state predictions across the four scenarios. Columns show the initial scene, MuJoCo final state, PIM prediction, and per-vertex error. Red arrows indicate initial external forces, and blue arrows indicate initial velocities.

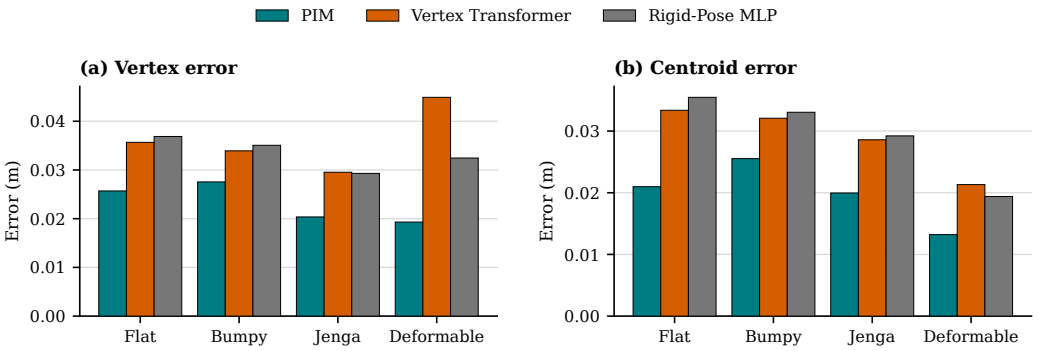


Figure 4: Comparison with the direct equilibrium-state prediction baselines across the four scenarios. The two panels report mean vertex error and mean centroid error; lower values indicate more accurate equilibrium-state prediction. Exact numerical values and edge errors are provided in Appendix Table 10.

protocol. All three methods degrade relative to matched ID, showing that this unseen geometry is challenging. PIM nevertheless retains the lowest vertex and centroid errors (Figure 5), obtaining $0.036957\,\mathrm{m}$ and $0.032656\,\mathrm{m}$, respectively, compared with $0.041459\,\mathrm{m}$ and $0.036174\,\mathrm{m}$ for Vertex Transformer and $0.045558\,\mathrm{m}$ and $0.046572\,\mathrm{m}$ for Rigid-Pose MLP. Relative to the strongest direct baseline, PIM reduces vertex and centroid errors by 10.9% and 9.7%, respectively. Although the numerical gaps in these scene-averaged metrics are moderate, Appendix Figure 8 shows visually pronounced differences on the unseen cube: PIM more closely preserves its terminal pose and geometry, whereas the two baselines exhibit conspicuous placement, orientation, or localized-geometry errors. Appendix Figure 7 provides corresponding qualitative comparisons for the position shifts.

Controlled Flat equilibrium-state comparison with rollout baselines. We retrain all controlled models on the same 1,000/100/100 Flat split and evaluate the same 100 test scenes at the exact $2\,\mathrm{s}$ equilibrium state on 642-vertex meshes. The trajectories are recorded every $0.008\,\mathrm{s}$ and contain 250 physical transitions. PIM and Small-PIM make one direct equilibrium-state prediction, HOPNet

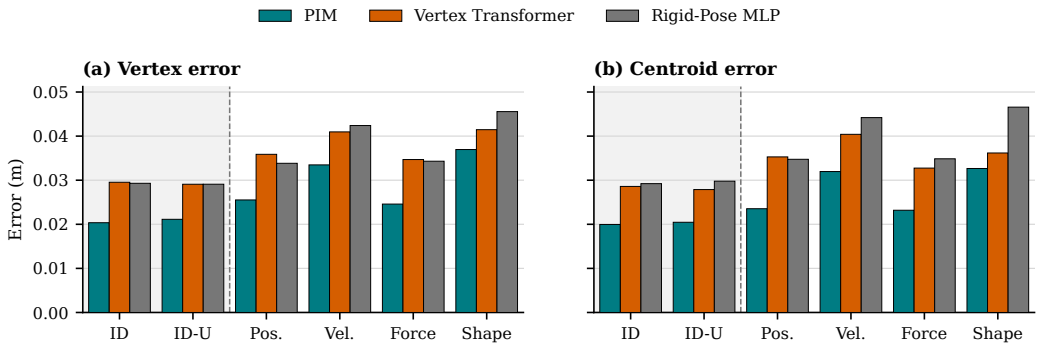


Figure 5: Jenga initial-condition and shape generalization without fine-tuning. The panels report mean vertex and centroid errors; lower is better. Exact values, including edge errors, are reported in Appendix Table 11.

Table 1: Controlled Flat equilibrium-state comparison at $t = 2\,\mathrm{s}$. All methods are evaluated on the same 100 test scenes. Errors are in meters and lower is better. Reimpl. denotes a PyTorch reimplementation developed for this comparison. † Triangle-edge errors are constrained by a rigid output, per-step rigid projection, or a rigid transform fitted to the coarse terminal prediction and transferred to the evaluation mesh.

Method	Parameters	$\varepsilon_{\mathrm{vtx}}$	$\varepsilon_{\mathrm{cen}}$	$\varepsilon_{\mathrm{tri}}$
PIM	19,755,622	**0.030731**	**0.027970**	9.03×10^{-6}
Small-PIM	5,013,302	0.031991	0.029233	9.79×10^{-6}
HOPNet	707,078	0.049890	0.048495	6.44×10^{-9} †
HOPNet-Adapt	707,462	0.038674	0.037202	6.44×10^{-9} †
FIGNet-Reimpl.	5,560,454	0.079846	0.078385	7.13×10^{-9} †
FIGNet-Reimpl.-Adapt	5,561,222	0.043898	0.041480	7.17×10^{-9} †
MeshGraphNet-Reimpl.	2,878,723	0.197970	0.194698	7.90×10^{-9} †
MeshGraphNet-Reimpl.-Adapt	2,879,107	0.219304	0.215845	8.42×10^{-9} †

performs 249 autoregressive updates after consuming its required input history, and the FIGNet and MeshGraphNet reimplementations perform 250 updates from $t = 0$. The physical scenes and terminal targets are shared, but each method retains its own preprocessing, state representation, and training supervision.

HOPNet-Adapt has the lowest vertex error among the autoregressive rollout baselines, while PIM and Small-PIM reduce its vertex error by 20.5% and 17.3%, respectively. The mean vertex-error difference between HOPNet-Adapt and PIM is $0.007943\,\mathrm{m}$. Appendix Figure 10 provides a qualitative comparison of the terminal configurations and the intermediate states produced by the autoregressive rollout models.

4.3 Physical Validity and Ablations

PIM achieves the highest valid-pair rate, penetration-free-scene rate, and contact F_1, as well as the lowest overlap rate, among the evaluated methods (Table 2). Rigid-Pose MLP has the smallest rigid-fit residual because rigidity is enforced by its output parameterization. The scene-level criterion is stricter than the pair-level criterion: a scene is counted as valid only when every object pair passes, so PIM's 94.83% valid-pair rate coexists with only 10.90% penetration-free scenes. The absolute penetration-free rate and contact F_1 (13.16%) remain modest, indicating that accurate equilibrium-state prediction alone does not guarantee a physically valid contact arrangement. These diagnostics concern only terminal inter-object geometry (Appendix D). We additionally evaluate a Flat-specific analytic non-penetration regularizer; its formulation and complete results are provided in Appendix F.

Table 2: Equilibrium-state validity on 1,000 held-out Flat scenes. Penetration and contact thresholds are 1 mm.

Method	Rigid (mm)	Valid pairs (%)	Overlap (%)	Valid scenes (%)	Contact F_1 (%)
PIM	0.035	**94.83**	**0.668**	**10.90**	**13.16**
Vertex Transformer	0.097	93.43	1.857	6.50	11.53
Rigid-Pose MLP	**0.017**	93.13	2.247	3.50	11.28

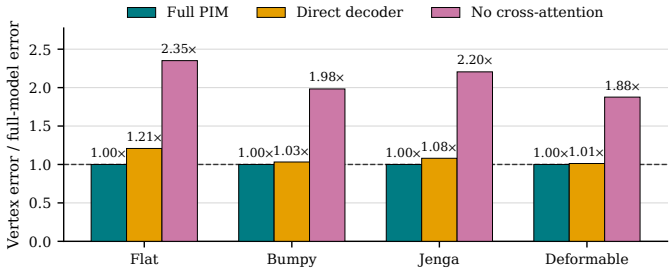


Figure 6: Controlled ablation results normalized by the full-model vertex error in each scenario. Full PIM is the unit reference. Exact vertex, centroid, and edge errors are provided in Appendix Table 12.

Removing cross-object attention increases vertex error by 87.6%–135.1% across the four scenarios (Figure 6; complete values are given in Appendix Table 12), confirming that object-internal processing alone is insufficient to recover multi-object interaction outcomes. Its centroid error increases by 109.2%–202.4%. On the three rigid scenarios, removing cross-object attention can nevertheless reduce edge error because this metric is invariant to global rigid motion and does not measure terminal object placement. Replacing the dual-branch decoder with a direct per-vertex decoder increases vertex error by 20.8%, 3.2%, 8.1%, and 1.3% on Flat, Bumpy, Jenga, and Deformable, respectively, and increases centroid error in all four scenarios. The explicit separation of object translation and relative vertex motion therefore provides a consistent placement advantage in these single-run comparisons. The direct decoder has lower edge error on Jenga, but its higher vertex and centroid errors show that improved local shape preservation does not imply more accurate terminal placement.

Edge-loss ablation. On Deformable, removing the edge loss increases vertex and centroid errors by only 4.2% and 5.7%, respectively, but raises the KNN edge-length error to $3.27\times$ its original value. Thus, aggregate placement errors mask a substantial loss of local geometric fidelity. Appendix Figure 9 shows the corresponding visible shape distortions, while complete values are reported in Appendix Table 12.

Hyperparameter sensitivity. On Deformable, we vary k and a separately over $\{16, 32, 64\}$ from the default $(32, 32)$ while retaining the same deterministic FPS rule. Vertex-error changes remain within 1.18%: using $k = 16$ reduces the measured fine-stage step time by 21.7% with slightly higher error, whereas using $k = 64$ increases step time by 45.4% for only a 0.23% reduction in vertex error. Changing a affects all errors by at most 0.71% and step time by at most 1.3%, with no monotonic improvement over the default. Complete results are provided in Appendix Table 13.

Runtime. Table 3 compares PIM with MuJoCo across scenarios and all learned models on the controlled Flat equilibrium-state task; the matched 2 s equilibrium-state protocol and the supplementary 100/102-step rollout timing protocol are detailed in Appendix D.3.

In the task-aligned controlled Flat terminal panel of Table 3, PIM provides low-latency prediction of object interaction outcomes relative to the evaluated rollout implementations, with $53.4\times$ lower latency and $32.8\times$ higher throughput than the fastest implementation without the additional force input, MeshGraphNet-Reimpl. Small-PIM has $49.3\times$ lower latency and $51.8\times$ higher throughput than the same rollout model. These measurements characterize implementation-specific inference

Table 3: Runtime results under the protocols in Appendix D.3. The first panel reports single-scene latency and matched parallel throughput across scenarios; the second reports the controlled Flat terminal task at 2 s.

	Single-scene latency			**Matched parallel throughput**		
Scenario	PIM (ms)	MuJoCo (ms)	Speedup	PIM (samples/s)	MuJoCo (samples/s)	Speedup
Flat	67.400	542.579	8.05×	16.779	11.347	1.48×
Bumpy	68.110	5,083.775	74.64×	16.868	1.490	11.32×
Jenga	66.383	1,686.671	25.41×	44.318	4.013	11.04×
Deformable	27.352	265,110.559	9,692.51×	64.102	0.0202	3,173.79×

Controlled Flat terminal timing at 2 s

Method	Calls	Latency $B/W{=}1$ (s)	Throughput $B/W{=}8$ (samples/s)
PIM	1	**0.0697**	25.9775
Small-PIM	1	0.0754	**41.0214**
HOPNet	249	461.6632	0.008279
HOPNet-Adapt	249	406.7234	0.008895
FIGNet-Reimpl.	250	8.3254	0.249863
FIGNet-Reimpl.-Adapt	250	9.6124	0.243478
MeshGraphNet-Reimpl.	250	3.7183	0.792329
MeshGraphNet-Reimpl.-Adapt	250	3.5632	0.777380

cost rather than hardware-normalized algorithmic complexity. PIM variants make one direct terminal prediction, whereas the autoregressive methods execute every update and additionally produce intermediate states.

5 Discussion and Limitations

When only the final configuration is required, PIM complements trajectory simulators by avoiding autoregressive rollout and directly predicting vertex-level outputs for rigid and deformable objects. Ablations show that cross-object communication is essential, center-relative decoding provides smaller scenario-dependent gains, and edge supervision reveals local shape degradation missed by vertex and centroid errors.

Evaluation is limited to scenario-specific models with fixed object correspondences and mesh topologies, a narrow range of deformable materials, and deformation magnitudes covered by the training distributions. Generalization to different object counts, mesh topologies, and larger deformations therefore remains open. Future work will investigate more general contact- and collision-aware objectives, explicit environment representations, and joint multi-scenario training.

6 Conclusion

We introduced PIM, a low-latency physical intuition model that predicts object interaction outcomes as terminal meshes from initial geometry, kinematics, external forces, and physical attributes. Local KNN attention captures intra-object geometry, FPS-anchor-based cross-object attention communicates interaction information, and a dual-branch decoder separates object translation from relative vertex motion. Across four rigid and deformable MuJoCo scenarios, PIM achieves lower terminal-state errors than the evaluated learned baselines. The ablations support the importance of cross-object interaction and show that edge supervision preserves local geometric fidelity in the Deformable scenario. Physical-validity diagnostics nevertheless expose a remaining gap between geometric prediction accuracy and physically valid configurations.

AI use statement

Generative AI tools were used for literature retrieval and organization; research ideation and critical examination of the methodology, experimental and evaluation protocols, implementation choices,

and interpretation of results; assistance with code and workflows used to generate the synthetic datasets; and manuscript drafting, reorganization, language editing, and LaTeX formatting. References suggested through this process were independently assessed against the original sources before inclusion, and AI-assisted code and workflows were reviewed and tested by the authors. Generative AI tools did not directly generate the reported simulation outcomes, model predictions, or numerical measurements, and no other required-disclosure uses apply to this work. All reported experimental data and measurements were produced by the simulation, training, and evaluation procedures described in the paper. The authors made the final methodological and editorial decisions, reviewed all AI-assisted content, and take full responsibility for the claims, citations, code, and experimental findings presented in this work.

Ethics statement

This work uses synthetic physical-interaction data generated in MuJoCo and does not involve human subjects, personal data, or user studies. We are not aware of direct ethical risks specific to the presented datasets or evaluation procedures.

Reproducibility statement

The main paper defines the task, model components, objectives, datasets, baselines, and principal evaluation metrics. Appendix A provides additional implementation details; Appendix B documents simulation, filtering, preprocessing, and data splits; Appendix C gives model, optimization, baseline, and ablation settings; and Appendix D specifies the complete evaluation and runtime protocols. The anonymized code repository is available at `https://anonymous.4open.science/r/supplementary_code-1C87/`.

References

Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: a framework for efficiently scaling neural operators. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Kelsey R Allen, Tatiana Lopez Guevara, Yulia Rubanova, Kim Stachenfeld, Alvaro Sanchez-Gonzalez, Peter Battaglia, and Tobias Pfaff. Graph network simulators can learn discontinuous, rigid contact dynamics. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1157–1167. PMLR, 14–18 Dec 2023a. URL `https://proceedings.mlr.press/v205/allen23a.html`.

Kelsey R Allen, Yulia Rubanova, Tatiana Lopez-Guevara, William F Whitney, Alvaro Sanchez-Gonzalez, Peter Battaglia, and Tobias Pfaff. Learning rigid dynamics with face interaction graph networks. In *The Eleventh International Conference on Learning Representations*, 2023b. URL `https://openreview.net/forum?id=J7Uh781A05p`.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 4509–4517, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Daniel M Bear, Elias Wang, Damian Mrowca, Felix J Binder, Hsiao-Yu Fish Tung, RT Pramod, Cameron Holdaway, Sirui Tao, Kevin Smith, Fan-Yun Sun, et al. Physion: Evaluating physical prediction from vision in humans and machines. *arXiv preprint arXiv:2106.08261*, 2021.

Shuhao Cao. Choose a transformer: fourier or galerkin. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: fast and memory-efficient exact attention with io-awareness. In *Proceedings of the 36th International*

Conference on Neural Information Processing Systems, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Saurabh Deshpande, Stéphane P.A. Bordas, and Jakub Lengiewicz. Magnet: A graph u-net architecture for mesh-based simulations. *Engineering Applications of Artificial Intelligence*, 133:108055, 2024. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2024.108055. URL https://www.sciencedirect.com/science/article/pii/S0952197624002136.

Paul Garnier, Vincent Lannelongue, Jonathan Viquerat, and Elie Hachem. Training transformers for mesh-based simulations. *Neural Comput. Appl.*, 38(11), June 2026. ISSN 0941-0643. doi: 10.1007/s00521-025-11731-3. URL https://doi.org/10.1007/s00521-025-11731-3.

Jiaqi Han, Wenbing Huang, Hengbo Ma, Jiachen Li, Joshua B. Tenenbaum, and Chuang Gan. Learning physical dynamics with subequivariant graph neural networks. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022.

Pan He, Patrick Emami, Sanjay Ranka, and Anand Rangarajan. Learning scene dynamics from point cloud sequences. *Int. J. Comput. Vision*, 130(3):669–695, March 2022. ISSN 0920-5691. doi: 10.1007/s11263-021-01551-y. URL https://doi.org/10.1007/s11263-021-01551-y.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (eds.), *Computer Vision – ECCV 2016*, pp. 646–661, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46493-0.

Zizhou Huang, Davi Colli Tozoni, Arvi Gjoka, Zachary Ferguson, Teseo Schneider, Daniele Panozzo, and Denis Zorin. Differentiable solver for time-dependent deformation problems with contact. *ACM Trans. Graph.*, 43(3), May 2024. ISSN 0730-0301. doi: 10.1145/3657648. URL https://doi.org/10.1145/3657648.

Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 430–438. JMLR.org, 2016.

Yifei Li, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. Diffcloth: Differentiable cloth simulation with dry frictional contact. *ACM Trans. Graph.*, 42(1), October 2022. ISSN 0730-0301. doi: 10.1145/3527660. URL https://doi.org/10.1145/3527660.

Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *ICLR*, 2019.

Zhehao Li, Qingyu Xu, Xiaohan Ye, Bo Ren, and Ligang Liu. Difffr: Differentiable sph-based fluid-rigid coupling for rigid body control. *ACM Trans. Graph.*, 42(6), December 2023a. ISSN 0730-0301. doi: 10.1145/3618318. URL https://doi.org/10.1145/3618318.

Zijie Li, Dule Shu, and Amir Barati Farimani. Scalable transformer for pde surrogate modeling. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023b. Curran Associates Inc.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Andreas Mayr, Sebastian Lehner, Arno Mayrhofer, Christoph Kloss, Sepp Hochreiter, and Johannes Brandstetter. Boundary graph neural networks for 3d simulations. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i8.26092. URL https://doi.org/10.1609/aaai.v37i8.26092.

Cheol-Hui Min, Jinseok Bae, Junho Lee, and Young Min Kim. GATSBI: Generative Agent-centric Spatio-temporal Object Interaction . In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3073–3082, Los Alamitos, CA, USA, June 2021. IEEE Computer Society. doi: 10.1109/CVPR46437.2021.00309. URL https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.00309.

Akihiro Nakano, Masahiro Suzuki, and Yutaka Matsuo. Interaction-based disentanglement of entities for object-centric world models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=JQc2VowqCzz.

Rhys Newbury, Jack Collins, Kerry He, Jiahe Pan, Ingmar Posner, David Howard, and Akansel Cosgun. A review of differentiable simulators. *IEEE Access*, 12:97581–97604, 2024. URL https://api.semanticscholar.org/CorpusID:271051266.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks, 2021. URL https://arxiv.org/abs/2010.03409.

Luis S. Piloto, Ari Weinstein, Peter W. Battaglia, and Matthew M. Botvinick. Intuitive physics learning in a deep-learning model inspired by developmental psychology. *Nature Human Behaviour*, 6:1257 – 1267, 2022. URL https://api.semanticscholar.org/CorpusID:250453635.

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. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Karl Schmeckpeper, Georgios Georgakis, and Kostas Daniilidis. Object-centric video prediction without annotation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13604–13610, 2021. doi: 10.1109/ICRA48506.2021.9561541.

Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Yeon-Ji Song, Jaein Kim, Suhyung Choi, Jin-Hwa Kim, and Byoung-Tak Zhang. Ock: Unsupervised dynamic video prediction with object-centric kinematics. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11359–11368, 2025. doi: 10.1109/ICCV51701.2025.01057.

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*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 32–42, 2021. doi: 10.1109/ICCV48922.2021.00010.

Tapas Tripura and Souvik Chakraborty. Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems. *Computer Methods in Applied Mechanics and Engineering*, 404:115783, 2023. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2022.115783. URL https://www.sciencedirect.com/science/article/pii/S0045782522007393.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40):eabi8605, 2021.

Nicholas Watters, Andrea Tacchetti, Théophane Weber, Razvan Pascanu, Peter Battaglia, and Daniel Zoran. Visual interaction networks: learning a physics simulator from video. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 4542–4550, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

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.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: a fast transformer solver for pdes on general geometries. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Zipeng Xiao, Zhongkai Hao, Bokai Lin, Zhijie Deng, and Hang Su. Improved operator learning by orthogonal attention. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Ce Yang, Weihao Gao, Di Wu, and Chong Wang. Learning to simulate unseen physical systems with graph neural networks. In *NeurIPS 2021 AI for Science Workshop*, 2021. URL https://openreview.net/forum?id=9Xh1v7Y9MAj.

Zongyao Yi. FIGNet PyTorch implementation, September 2024. URL https://github.com/jongyaoY/fignet.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. In *Proceedings - 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021*, Proceedings of the IEEE International Conference on Computer Vision, pp. 16239–16248. Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/ICCV48922.2021.01595. Publisher Copyright: © 2021 IEEE; 18th IEEE/CVF International Conference on Computer Vision, ICCV 2021 ; Conference date: 11-10-2021 Through 17-10-2021.

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 (CVPR)*, June 2019.

A Method Details

A.1 Residual Attention Implementation

The local graph has effective neighborhood size $k_i = \min(k, n_i - 1)$ and is constructed once from $\mathbf{X}_i^0$. It is reused by all local-attention blocks and the edge-length objective. Local-attention and feed-forward outputs use residual connections with learnable LayerScale vectors (Touvron et al., 2021) and DropPath (Huang et al., 2016):

$$\widetilde{\mathbf{H}}_i = \mathbf{H}_i + \mathrm{DropPath}(\gamma_a \odot \mathrm{LocalSA}(\mathbf{H}_i)), \tag{12}$$

$$\mathbf{H}_i \leftarrow \widetilde{\mathbf{H}}_i + \mathrm{DropPath}\Big(\gamma_f \odot \mathrm{FFN}(\widetilde{\mathbf{H}}_i)\Big). \tag{13}$$

The feed-forward block uses a gated SwiGLU-style structure (Shazeer, 2020).

Cross-object messages are incorporated through

$$\widetilde{\mathbf{H}}_i = \mathbf{H}_i + \mathrm{DropPath}(\gamma_c \odot W_o \mathbf{Z}_i^{\mathrm{cross}}). \tag{14}$$

Each cross-object update is followed by local attention and feed-forward refinement.

B Dataset Construction

B.1 Simulation Pipeline

All datasets are generated in SI units with MuJoCo (Todorov et al., 2012). Rigid objects are represented by free-joint bodies, while the deformable object is represented by a MuJoCo flex body

Table 4: MuJoCo rigid-geometry parameter presets. Density is given in $\mathrm{kg/m^3}$; sliding friction μ^t is dimensionless, while torsional and rolling coefficients μ^s and μ^r have units of length. The coefficients are simulation parameters intended to span diverse contact behavior rather than calibrated measurements of specific material pairs.

Material	ρ ($\mathrm{kg/m^3}$)	$\boldsymbol{\mu}$	Material	ρ ($\mathrm{kg/m^3}$)	$\boldsymbol{\mu}$
HDPE	950	$(0.22, 0.012, 0.012)$	Rubber	1100	$(0.55, 0.020, 0.012)$
Pine	500	$(0.35, 0.018, 0.012)$	Oak	700	$(0.40, 0.020, 0.012)$
Ice	917	$(0.10, 0.006, 0.006)$	Glass	2500	$(0.25, 0.012, 0.012)$
Concrete	2400	$(0.50, 0.020, 0.012)$	Aluminum	2700	$(0.35, 0.018, 0.012)$

Table 5: Dataset statistics for the four scenarios. Vertex counts are totals per sample, with object-wise composition in parentheses. Coarse meshes use a fixed subsampling stride of four.

Scenario	Objects	Train	Val.	Test	Fine vertices / sample	Coarse vertices / sample
Flat	9	30,000	1,000	1,000	5,778 (9×642)	1,449 (9×161)
Bumpy	9	30,000	1,000	1,000	5,778 (9×642)	1,449 (9×161)
Jenga	10	40,000	2,000	2,000	2,116 $(2 \times 642 + 8 \times 104)$	530 $(2 \times 161 + 8 \times 26)$
Deformable	3	5,000	200	200	1,577 $(2 \times 642 + 293)$	396 $(2 \times 161 + 74)$

with vertex-level degrees of freedom. We use a $0.002\,\mathrm{s}$ time step, implicit integration, elliptic friction cones, viscosity 6×10^{-4}, friction-to-normal constraint impedance ratio $\texttt{impratio} = 5$, 200 solver iterations, and tolerance 10^{-10}. Contact parameters are $\mathrm{solref} = (0.005, 1)$ and $\mathrm{solimp} = (0.90, 0.97, 0.02)$, with joint damping 0.01. Flat, Bumpy, and Jenga use contact dimensionality $\texttt{condim} = 6$, enabling sliding, torsional, and rolling friction, whereas Deformable uses $\texttt{condim} = 3$, which includes tangential sliding friction but not torsional or rolling friction.

For Flat, Bumpy, and Deformable, gravity is sampled as $g_z = -9.81(1+\epsilon)$ with $\epsilon \sim \mathcal{U}[-0.05, 0.05]$; Jenga uses fixed gravity $g_z = -9.81\,\mathrm{m/s^2}$. External forces are applied during the first $0.05\,\mathrm{s}$. Simulation terminates at $10\,\mathrm{s}$ for Flat, Bumpy, and Jenga and at $8\,\mathrm{s}$ for Deformable, or earlier when the maximum absolute generalized velocity remains below 10^{-3} for $0.2\,\mathrm{s}$.

Flat and Bumpy sample rigid-object materials uniformly from HDPE, rubber, pine, and oak. The Jenga impactors and the two Deformable impactors use the complete eight-material pool in Table 4, while the Jenga tower blocks are fixed to pine. Sliding, torsional, and rolling friction are independently perturbed by $\pm 10\%$, $\pm 20\%$, and $\pm 20\%$, respectively, and clamped to $[0.12, 0.60]$, $[0.005, 0.020]$, and $[0.003, 0.012]$. The supporting surface uses fixed HDPE parameters.

B.2 Splits and Preprocessing

Training, validation, and test splits use the same scenario-specific sampling distributions but disjoint random-number streams. Validation and test generation additionally use split-specific random seeds for individual simulation attempts, preventing accepted samples from being shared across splits. Table 5 summarizes the resulting split sizes and mesh resolutions.

For the controlled comparison, the Flat split contains 1,000/100/100 training/validation/test scenes. Each trajectory is recorded every $0.008\,\mathrm{s}$ and terminates at $2\,\mathrm{s}$, yielding 250 physical transitions in addition to the two preceding frames required to initialize HOPNet. Scene identifiers, split membership, and underlying physical trajectories are shared. PIM and Small-PIM use separately preprocessed 642-vertex initial–terminal mesh pairs. HOPNet constructs its method-specific collision-complex representation and uses all 250,000 training and 25,000 validation transitions per epoch. FIGNet-Reimplemented and MeshGraphNet-Reimplemented independently construct their respective 64-node graphs and sample 16 transitions per training scene and 16 fixed transitions per validation scene, yielding 16,000 training and 1,600 validation examples per epoch.

Table 6: Scenario-specific initial-state ranges shown component-wise. For Flat/Bumpy objects F–I, the signs of the planar velocities are selected to point toward the interaction region.

Objects	Axis	Position (m)	Velocity $(\mathrm{m/s})$	Force (N)
Flat/Bumpy A	x	$[-.04, .04]$	$[-.01, .01]$	$[-6, 6]$
	y	$[-.04, .04]$	$[-.01, .01]$	$[-6, 6]$
	z	$[.10, .14]$	0	0
Flat/Bumpy B–E	x	$\pm[.125, .175]$	$[-.01, .01]$	$[-6, 6]$
	y	$\pm[.125, .175]$	$[-.01, .01]$	$[-6, 6]$
	z	$[.10, .14]$	0	0
Flat/Bumpy F–I	x	$\pm[.25, .4]$	± 1.3	$[-6, 6]$
	y	$\pm[.25, .4]$	± 1.3	$[-6, 6]$
	z	$[.10, .14]$	0	0
Jenga A	x	$[-.45, -.38]$	$[2.8, 3.5]$	$[10, 14]$
	y	$[-.075, -.045]$	$[-.03, .03]$	$[-.4, .4]$
	z	$[.05, .08]$	$[.05, .25]$	$[-.2, .2]$
Jenga B	x	$[.38, .45]$	$[-3.5, -2.8]$	$[-14, -10]$
	y	$[.045, .075]$	$[-.03, .03]$	$[-.4, .4]$
	z	$[.14, .17]$	$[.2, .5]$	$[-.2, .2]$
Deformable A	x	$[-.12, .12]$	$[-.25, .25]$	$[-1, 1]$
	y	$[-.30, -.16]$	$[.8, 2.0]$	$[2, 8]$
	z	$[.05, .09]$	$[-.1, .2]$	$[-.5, .5]$
Deformable B	x	$[-.12, .12]$	$[-.25, .25]$	$[-1, 1]$
	y	$[.16, .30]$	$[-2.0, -.8]$	$[-8, -2]$
	z	$[.05, .09]$	$[-.1, .2]$	$[-.5, .5]$

B.3 Scenario Details

Table 6 summarizes the initial-state distributions. Intervals denote independent uniform samples; the signs of the Flat/Bumpy corner velocities are selected to point toward the interaction region.

Flat. Nine spheres of radius $0.05\,\mathrm{m}$ interact on a horizontal HDPE surface. Objects A–E initialize within or around the central interaction region, while F–I enter from four outer quadrants. Variations in material, planar force, gravity, and friction produce collision, sliding, rolling, and stabilization outcomes.

Bumpy. Bumpy retains the same sphere distributions but replaces the plane by a fixed 513×513 height field over $[-0.98, 0.98]^2\,\mathrm{m}^2$. Its height ranges from -0.01091 to $0.01130\,\mathrm{m}$ and combines $0.01\,\mathrm{m}$ base perturbations with 12 pits of depth $0.005\,\mathrm{m}$ and a $0.04\,\mathrm{m}$ boundary fade. The terrain is generated once with a random seed. Flat and Bumpy use separate models.

Jenga. The tower contains eight stationary pine blocks in four alternating layers centered at heights 0.025, 0.075, 0.125, and $0.175\,\mathrm{m}$. Each layer contains two blocks whose centers are offset by $\pm 0.055\,\mathrm{m}$ along the horizontal axis perpendicular to the blocks' long direction. Alternating block dimensions are $0.280 \times 0.070 \times 0.050\,\mathrm{m}^3$ and $0.070 \times 0.240 \times 0.050\,\mathrm{m}^3$. Two radius-$0.05\,\mathrm{m}$ impactors, whose materials are sampled from Table 4, approach from opposite sides.

Deformable. Two radius-$0.05\,\mathrm{m}$ rigid spheres approach a stationary flex box from opposite sides. The box uses full vertex degrees of freedom on a $13 \times 6 \times 6$ lattice spanning $0.34 \times 0.14 \times 0.12\,\mathrm{m}^3$, with total mass $0.5\,\mathrm{kg}$, flex radius $0.009\,\mathrm{m}$, friction $(1.0, 0.04, 0.02)$, and edge constraints with $\mathrm{solref} = (0.022, 1)$ and $\mathrm{solimp} = (0.92, 0.98, 0.01)$. Its exported surface contains 293 vertices and 580 triangular faces.

For Flat, Bumpy, and Jenga, an object pair is counted as impacted after a three-step warm-up when it is in contact with relative speed at least $0.03\,\mathrm{m/s}$ and either normal force at least $20\,\mathrm{N}$ or detected

Table 7: Jenga initial-condition ranges used for the generalization evaluation. Intervals give the absolute x-component magnitudes for the two oppositely directed impactors. Only the listed factor changes in each single-factor OOD set.

Factor	ID range	Lower-range OOD	Upper-range OOD
Initial position $\lvert x\rvert$ (m)	$[0.38, 0.45]$	Close $[0.30, 0.37]$	Far $[0.46, 0.53]$
Initial speed $\lvert v_x\rvert$ (m/s)	$[2.8, 3.5]$	Slow $[2.0, 2.7]$	Fast $[3.6, 4.3]$
Applied force $\lvert F_x\rvert$ (N)	$[10.0, 14.0]$	Low $[5.5, 9.5]$	High $[14.5, 18.5]$

penetration. Flat retains scenes with at least eight of the 36 sphere pairs impacted, whereas Bumpy requires at least five impacted pairs. Jenga monitors the 16 impactor–block pairs and requires at least two impacted pairs, at least one impact involving each impactor, and tower collapse. Collapse requires at least three blocks to translate horizontally by more than $0.05\,\mathrm{m}$ or tilt by more than 30°, including a top-layer block, or a top-layer block center to fall below $0.10\,\mathrm{m}$. Non-finite Jenga states are rejected, as are samples in which a tower block exceeds $0.60\,\mathrm{m}$ total displacement, $0.55\,\mathrm{m}$ horizontal displacement, or the height interval $[-0.02, 0.40]\,\mathrm{m}$.

Deformable candidates are retained only when at least one rigid sphere contacts the flex body and the maximum soft-vertex displacement reaches $0.002\,\mathrm{m}$; initial rigid–soft contact is rejected. These event-conditioned filters prevent the retained datasets from being dominated by non-interacting or numerically failed trials while preserving variation in terminal outcomes.

B.4 Jenga Initial-Condition and Shape Generalization

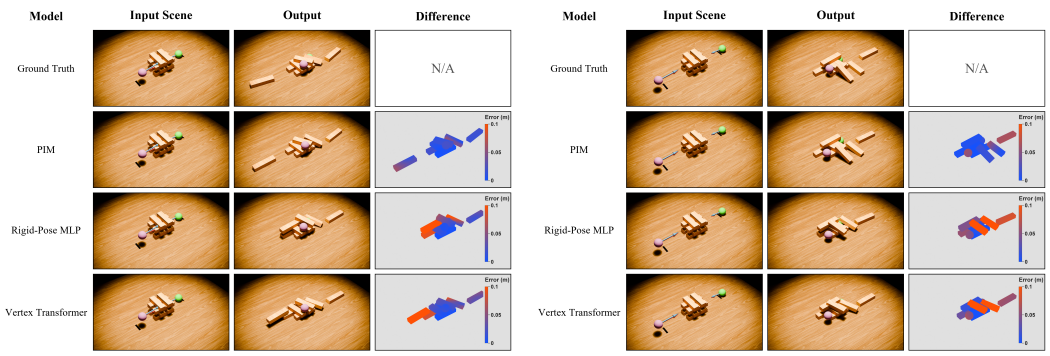
We evaluate the fixed validation-selected Jenga checkpoints without retraining or fine-tuning. Table 7 gives the original and shifted initial-condition ranges. Because the two impactors approach from opposite sides, the table reports absolute magnitudes of their x components. Each single-factor initial-condition OOD set contains 2,000 accepted scenes, with 900 matched-protocol and 100 outcome-unfiltered scenes from each of the lower- and upper-range branches. The matched protocol retains the original impact and tower-collapse requirements. The outcome-unfiltered protocol removes only these two requirements while retaining the initial-overlap checks, numerical- and non-finite-state rejection, and the displacement and height bounds described above. ID-U contains 2,000 outcome-unfiltered scenes sampled from the original input ranges. For these sets, all non-varied coordinates, materials, friction parameters, gravity, tower geometry, simulation horizon, and the $0.05\,\mathrm{s}$ force duration remain unchanged.

The separate shape-OOD set contains 2,000 matched-protocol scenes and changes only top-layer object J from its training-time elongated block to an equal-volume cube with side length $0.094354\,\mathrm{m}$. Both the visual mesh and MuJoCo collision geometry are changed to the cube. The cube retains the original object's 104 vertices, 204 triangular faces, vertex order, and face-index order, preserving the ten-object model interface while changing its geometry. Object I, the impactors, all initial-condition ranges, materials, friction parameters, gravity, simulation horizon, force duration, and event-conditioned acceptance criteria remain unchanged.

C Training and Baseline Details

C.1 PIM Training

Flat and Bumpy use a hidden dimension of 512, eight attention heads, two intra-object layers, and two interaction layers; Jenga and Deformable use a hidden dimension of 768, six attention heads, four intra-object layers, and two interaction layers. All configurations use KNN size 32, 32 FPS source anchors, and a feed-forward expansion factor of four. Coarse training on stride-four meshes is used as initialization rather than as inference-time multi-resolution feature fusion. After validation-based checkpoint selection, the best coarse checkpoint initializes the same architecture at fine resolution. The fine stage recomputes vertex features and KNN indices on the full-resolution meshes; coarse features are not concatenated with fine features or reused during fine-stage inference. Checkpoint selection in both stages uses validation coordinate L1 error.



(a) Close position OOD **(b) Far position OOD**

Figure 7: Qualitative comparison on matched-protocol Jenga position-OOD scenes. Each panel shows the shared input scene, the ground-truth equilibrium state or model prediction, and the per-vertex Euclidean error relative to the ground truth. All error maps use the same $[0, 0.1]$ m color scale. Red and blue arrows indicate the initial applied force and velocity, respectively. Short black line segments mark the boundary of the original ID position range adjacent to each OOD branch. For close OOD, the marker denotes $|x| = 0.38$ m and the impactors are sampled from $|x| \in [0.30, 0.37]$ m; for far OOD, it denotes $|x| = 0.45$ m and the impactors are sampled from $|x| \in [0.46, 0.53]$ m. These examples illustrate differences in terminal placement among PIM, Rigid-Pose MLP, and Vertex Transformer; aggregate performance is reported in Figure 5 and Table 11.

Table 8: Coarse and fine training configurations. Global batch size includes all samples processed across four GPUs and all gradient-accumulation steps.

Scenario	Stage	Batch/GPU	Accum.	Global	Local chunk	Cross chunk
Flat/Bumpy	Coarse	16	1	64	256	16
Flat/Bumpy	Fine	4	2	32	1024	16
Jenga/Deform.	Coarse	32	1	128	256	16
Jenga/Deform.	Fine	8	1	32	1024	16

All models use AdamW (Loshchilov & Hutter, 2019) with linear warm-up followed by cosine decay. Flat and Bumpy use weight decay 3×10^{-4} and dropout 0.1, whereas Jenga and Deformable use weight decay 10^{-4} and no dropout. All configurations use DropPath rate 0.1 and LayerScale initialization 10^{-3}.

The Flat and Bumpy coarse stages train for 80 epochs with learning rate 3×10^{-4}, minimum learning rate 10^{-5}, and a 4% warm-up. The Jenga and Deformable coarse stages use the same learning-rate schedule for 100 epochs. All coarse stages use $\lambda_{\text{cen}} = 1$ and $\lambda_{\text{edge}} = 5$, with the edge weight increased linearly from zero over the first 10 epochs.

Flat fine training runs for 40 epochs with learning rate 1.2×10^{-4}, minimum learning rate 10^{-5}, a 5% warm-up, $\lambda_{\text{cen}} = 0.1$, and $\lambda_{\text{edge}} = 20$. Bumpy fine training runs for 20 epochs with learning rate 2×10^{-5}, minimum learning rate 2×10^{-6}, a 15% warm-up, $\lambda_{\text{cen}} = 1$, and $\lambda_{\text{edge}} = 5$. Jenga and Deformable fine training run for 40 epochs using the Flat fine-stage learning-rate and loss settings. Per-stage batch sizes and chunk sizes are listed in Table 8.

Models are implemented in PyTorch and trained on four NVIDIA RTX A6000 GPUs using distributed data parallelism and mixed precision. All attention layers use PyTorch scaled dot-product attention with the FlashAttention backend (Dao et al., 2022). Gradient norm is clipped to 1.0. Local queries and object pairs are processed in chunks without changing neighborhoods or pairwise softmax normalization.

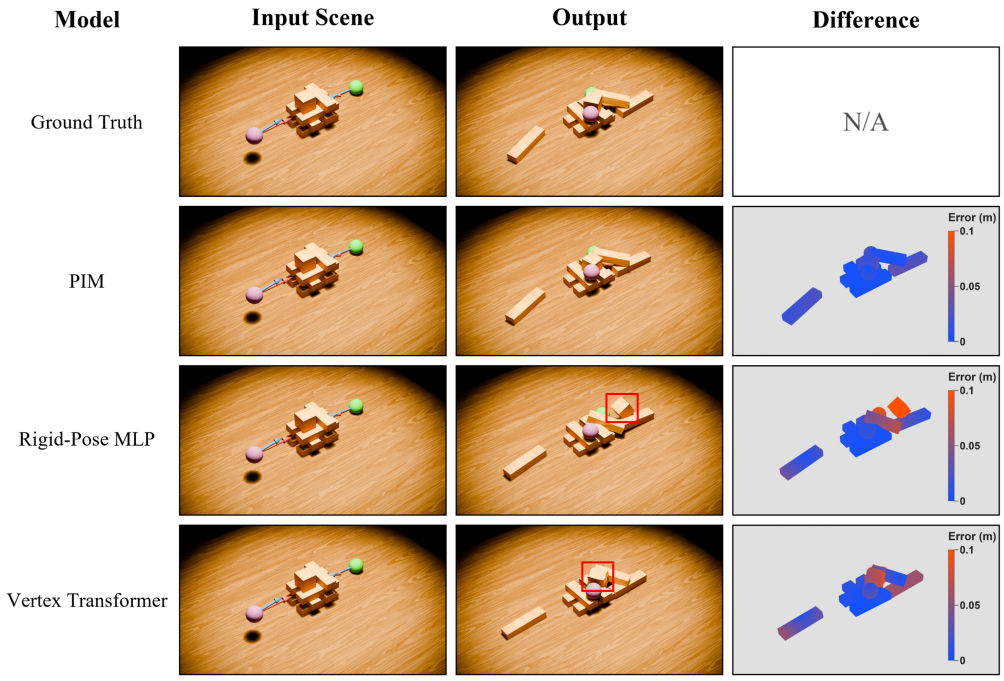


Figure 8: Qualitative comparison on a matched-protocol Jenga shape-OOD scene. Object J, one of the two top-layer blocks, is replaced by an equal-volume cube not observed as a Jenga tower block during training; both its visual mesh and collision geometry are cubical. Each row shows the shared input scene, the ground-truth equilibrium state or model prediction, and the per-vertex Euclidean error relative to the ground truth. All error maps use the same $[0, 0.1]\,\mathrm{m}$ color scale. Red and blue arrows indicate the initial applied force and velocity, respectively. The red boxes locate the unseen cube in the two baseline outputs and highlight their visible failure modes in this example: incorrect terminal placement or orientation for Rigid-Pose MLP and a larger localized geometry-and-pose error for Vertex Transformer. Aggregate results over all 2,000 shape-OOD scenes are reported in Figure 5 and Table 11.

C.2 Direct Equilibrium-State Baselines

Rigid-Pose MLP. Rigid-Pose MLP represents each object by a 30-dimensional geometric and physical descriptor containing its initial center, center relative to the scene center, axis-aligned extent, the root-mean-square distance of vertices from the object centroid, six upper-triangular covariance entries, initial velocity, applied force, density, three friction coefficients, and four scene-level conditions. A learned 16-dimensional embedding of each object's index in the fixed object ordering is concatenated to this descriptor. A shared layer-normalized MLP maps each object to a 256-dimensional embedding. Mean and max pooling over the object embeddings are concatenated and projected to a 256-dimensional scene context, which is broadcast back to every object. The object and scene embeddings are then concatenated and processed by one residual MLP block of width 256.

Two prediction heads output a three-dimensional center displacement and a continuous 6D rotation representation (Zhou et al., 2019). The latter is converted to a proper rotation matrix by orthogonalization. For object i, the predicted terminal vertices are

$$\widehat{\mathbf{Y}}_i = \left(\mathbf{X}_i^0 - \mathbf{c}_i^0\right)\mathbf{R}_i + \mathbf{c}_i^0 + \Delta\mathbf{c}_i. \tag{15}$$

The final translation and rotation layers are initialized to the identity transformation. Rigid-Pose MLP therefore predicts a rigid terminal pose for every object. It is evaluated on all four scenarios; on Deformable, it serves as a constrained rigid-output reference and cannot reproduce local deformation.

Vertex Transformer. Vertex Transformer is a hierarchical vertex-level equilibrium-state predictor developed for this study, rather than a reimplementation of the Point Transformer architecture of Zhao et al. (Zhao et al., 2021). Each vertex receives centered coordinates, RMS-normalized coordinates, normalized radius, and three sinusoidal/cosine Fourier-coordinate bands, producing a 25-dimensional geometric feature. The same 30-dimensional object descriptor used by Rigid-Pose MLP is concatenated with a learned 16-dimensional embedding of the object's index in the fixed object ordering and projected to the model width. The projected condition is added to every vertex token of the corresponding object.

The network uses hidden width 384, six attention heads, KNN size 32, and feed-forward expansion factor four. One intra-object vertex-attention block applies six-head self-attention over cached KNN neighborhoods. Embedded relative coordinate differences are added to the neighbor key and value features and also parameterize a learned per-head attention bias. Mean and max pooling then reduce each object to one token, and one standard Transformer encoder layer exchanges information among the object tokens. The resulting object context is projected and broadcast back to all vertices of the corresponding object. A zero-initialized MLP directly predicts a three-dimensional displacement for each vertex:

$$\widehat{\mathbf{Y}}_i = \mathbf{X}_i^0 + \mathrm{MLP}_{\mathrm{dec}}(\mathbf{H}_i). \tag{16}$$

Unlike PIM, it exchanges inter-object information only through pooled object tokens and does not use point-level cross-object attention or the center–relative dual-branch decoder.

Training protocol. Both direct equilibrium-state prediction baselines operate on the fine-resolution meshes and use the same training, validation, and test splits and the same equilibrium-state evaluator as PIM. Each model is trained independently for 100 epochs on one NVIDIA RTX A6000 with mixed precision, AdamW, initial learning rate 10^{-4}, weight decay 10^{-4}, a 5% linear warm-up, and cosine decay to 10^{-5}. Gradients are clipped to norm 1.0. Training uses the same loss terms and weights as the PIM fine stage,

$$\mathcal{L} = \mathcal{L}_{\mathrm{coord}} + 0.1\mathcal{L}_{\mathrm{cen}} + 20\mathcal{L}_{\mathrm{edge}}, \tag{17}$$

and checkpoints are selected by validation coordinate L1 error.

C.3 Controlled Flat Comparison

PIM variants. The controlled PIM retains the Flat architecture and objective used in the main experiments: hidden width 512, eight attention heads, two intra-object layers, two cross-object interaction layers, feed-forward expansion factor four, KNN size 32, and 32 source anchors. It has 19,755,622 trainable parameters. PIM is trained for 80 coarse epochs with initial learning rate 3×10^{-4} and then for 40 fine epochs with initial learning rate 1.2×10^{-4}. Small-PIM retains the input representation, KNN and anchor counts, both interaction layers, decoder, losses, and the same 80/40-epoch coarse-to-fine schedule. It uses hidden width 256, four attention heads, three pre-interaction intra-object layers, two interaction layers, and feed-forward expansion factor three, giving 5,013,302 trainable parameters.

HOPNet. HOPNet (Wei & Fink, 2025) is retrained using its released architecture and objective, with 128 latent channels, one graph-processing layer, and two-layer ReLU MLPs. Rather than using PIM's preprocessed initial–terminal mesh pairs, it constructs its method-specific collision-complex representation from every recorded transition. The complexes use an object radius of $0.1\,\mathrm{m}$ and a $0.01\,\mathrm{m}$ local face neighborhood, and retain the scene-specific table properties. HOPNet is trained for 40 epochs with Adam, a constant learning rate of 10^{-4}, and all 250,000 training transitions per epoch; the final-epoch checkpoint is used for evaluation. Its rollout consumes the required history and performs 249 learned updates to reach the exact $2\,\mathrm{s}$ target. HOPNet-Adapt appends the normalized external-force vector applied during the initial forcing interval to the object feature:

$$\mathbf{x}_{i,t,\mathrm{adapt}}^{(4)} = \left[\mathbf{x}_{i,t}^{(4)}, \mathbf{f}_{i,t}/10\right] \in \mathbb{R}^{16}. \tag{18}$$

The force is zero for the static floor, and all other model, target, graph, and optimization settings remain unchanged.

FIGNet-Reimplemented. FIGNet-Reimplemented is based on the released PyTorch implementation of FIGNet (Yi, 2024) while following the architecture and one-step objective described by the original method (Allen et al., 2023b). It is therefore our PyTorch reimplementation for the controlled Flat comparison rather than an execution of the official FIGNet source implementation. Each sphere is represented by 64 vertices, and the table is represented by explicit boxes. The model consumes three consecutive position frames, uses ten message-passing blocks of width 128 with two-layer ReLU MLPs, and predicts a normalized one-step position acceleration. Training uses Gaussian position noise with standard deviation $3 \times 10^{-5}\,\mathrm{m}$, gradient clipping at one, Adam with constant learning rate 10^{-4}, and 100 epochs. Each epoch samples 16 transitions per training scene and uses 16 fixed transitions per validation scene. During rollout, sphere–sphere and sphere–box face interactions are reconstructed from the predicted state using a $0.01\,\mathrm{m}$ surface margin. A least-squares rigid transform is applied after every predicted update and is fed back into the next step. FIGNet-Reimplemented-Adapt changes only the physical-property vector from 9 to 12 dimensions by appending the raw three-dimensional external-force vector.

MeshGraphNet-Reimplemented. Our PyTorch Geometric implementation adapts the MeshGraphNets architecture (Pfaff et al., 2021) to three-dimensional rigid spheres. It is our reimplementation for the controlled Flat comparison rather than an execution of an official rigid-body implementation; the codebase from which it was developed originally targeted two-dimensional cylinder flow. It uses 64 vertices per sphere, 15 message-passing blocks of width 128, static triangle edges, and dynamic inter-object world edges with radius $0.03\,\mathrm{m}$ and at most eight senders per receiver. The network predicts a normalized one-step vertex-velocity increment, after which positions are updated by semi-implicit Euler with $\Delta t = 0.008\,\mathrm{s}$. Training uses velocity noise with standard deviation $0.002\,\mathrm{m/s}$, Adam with constant learning rate 10^{-4}, 100 epochs, and the same 16 training and 16 fixed validation transitions per scene used by FIGNet-Reimplemented. No rigidity constraint is applied during rollout. A proper rigid transform is fitted only to the terminal 64-vertex output to transfer it to the common 642-vertex evaluation mesh, and this terminal bridge is never fed back into the trajectory. MeshGraphNet-Reimplemented-Adapt changes only the node feature from 14 to 17 dimensions by appending $\mathbf{f}_{i,t}/10$.

Data and comparison protocol. All models are trained on one RTX A6000. The scene manifests, split membership, physical trajectories, force records, and terminal evaluation meshes are shared, but the training data are not converted into a single common representation: PIM uses initial–terminal mesh pairs, HOPNet uses collision complexes, and the two graph-network reimplementations construct their own temporal graphs and one-step targets. Consequently, the comparison controls the underlying simulated scenes and terminal targets, but not the learning objective, number of supervised transitions, parameter count, or optimization cost.

HOPNet-Adapt is the strongest rollout baseline with vertex error $0.038674\,\mathrm{m}$ and centroid error $0.037202\,\mathrm{m}$, followed by FIGNet-Reimplemented-Adapt at $0.043898\,\mathrm{m}$, HOPNet at $0.049890\,\mathrm{m}$, FIGNet-Reimplemented at $0.079846\,\mathrm{m}$, and the two MeshGraphNet variants in terms of vertex error. Adding the interval force reduces HOPNet vertex and centroid errors by 22.5% and 23.3%, respectively. It reduces FIGNet vertex error by 45.0%, showing that explicit knowledge of the applied external force is useful for both architectures. The same modification increases MeshGraphNet vertex error by 10.8%, so providing the external-force input alone does not guarantee more accurate long-horizon rollout. In particular, the MeshGraphNet-Adapt rigid-fit residual of the 64-vertex prediction decreases from 0.3266 to $0.2788\,\mathrm{m}$ while its terminal vertex error increases from 0.1980 to $0.2193\,\mathrm{m}$; improved rigid fit of the coarse prediction therefore does not imply improved terminal placement. The near-zero triangle-edge errors of HOPNet and FIGNet arise from their rigid pose output or per-step rigid projection, and those of MeshGraphNet arise from the terminal rigid bridge. They should not be interpreted as evidence that these models produce more accurate terminal configurations. PIM avoids autoregressive accumulation by predicting the terminal mesh directly. Despite using 74.6% fewer parameters than PIM, Small-PIM increases vertex and centroid errors by only 4.1% and 4.5%, respectively. At a parameter count comparable to FIGNet-Reimplemented-Adapt, it also achieves a 27.1% lower vertex error, showing that the advantage of direct terminal prediction persists at substantially reduced model capacity.

C.4 Controlled Ablations

The no-cross-attention ablation removes only the cross-object update while retaining local attention, feed-forward refinement, residual connections, and the dual-branch decoder. The direct-decoder ablation replaces the two decoder branches by a zero-initialized per-vertex MLP. Both use the full coarse-to-fine schedule, losses, splits, and checkpoint selection. The Deformable no-edge-loss ablation sets $\lambda_{\text{edge}} = 0$ in both the coarse and fine stages while retaining the architecture, data splits, coordinate and centroid losses, optimizer, training schedule, and validation-based checkpoint selection. Its fine stage is initialized from the corresponding no-edge-loss coarse checkpoint.

C.5 Hyperparameter Sensitivity

We conduct the hyperparameter sensitivity study on Deformable, whose coarse representation contains 161 vertices per rigid sphere and 74 vertices for the soft box. Consequently, all evaluated values in $\{16, 32, 64\}$ correspond to distinct neighborhoods and source-anchor sets. Starting from the default $(k, a) = (32, 32)$, we vary one quantity at a time using $(16, 32)$, $(64, 32)$, $(32, 16)$, and $(32, 64)$. All variants retain the same architecture, parameter count, fixed data splits, coarse-to-fine schedule, optimizer, losses, and validation-based checkpoint criterion.

To decouple the local-attention neighborhood from geometric supervision, the edge loss and edge metric use a fixed 32-neighbor graph in every variant, while local attention uses the specified value of k. Each configuration is trained independently through the complete coarse-to-fine pipeline, and every fine-stage model is initialized from its matching coarse-stage checkpoint. The $(32, 32)$ reference is retrained as part of the sweep rather than reusing the main-result checkpoint. We report test-set results from this matched run. Relative training time is computed from the median synchronized duration of 20 profiled fine-stage training steps in the first epoch on four RTX A6000 GPUs and normalized by the matched $(32, 32)$ measurement.

D Evaluation Details

D.1 Geometric Metrics

Mean vertex error is

$$\varepsilon_{\text{vtx}} = \frac{1}{\sum_i n_i} \sum_{i=1}^{M} \sum_{p=1}^{n_i} \|\widehat{\mathbf{y}}_{i,p} - \mathbf{y}_{i,p}\|_2. \tag{19}$$

Mean centroid error is

$$\varepsilon_{\text{cen}} = \frac{1}{M} \sum_{i=1}^{M} \|\widehat{\mathbf{c}}_i - \mathbf{c}_i\|_2. \tag{20}$$

KNN edge-length error is

$$\varepsilon_{\text{edge}} = \frac{1}{\sum_i n_i k_i} \sum_{i,p} \sum_{q \in \mathcal{N}_i(p)} \left| \|\widehat{\mathbf{y}}_{i,p} - \widehat{\mathbf{y}}_{i,q}\|_2 - \|\mathbf{y}_{i,p} - \mathbf{y}_{i,q}\|_2 \right|. \tag{21}$$

For the controlled rollout comparisons, we additionally evaluate edge-length error on the shared triangle topology of the 642-vertex evaluation mesh. Let $\mathcal{E}_i^{\text{tri}}$ denote the unique triangle edges of object i. We define

$$\varepsilon_{\text{tri}} = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{|\mathcal{E}_i^{\text{tri}}|} \sum_{(p,q) \in \mathcal{E}_i^{\text{tri}}} \left| \|\widehat{\mathbf{y}}_{i,p} - \widehat{\mathbf{y}}_{i,q}\|_2 - \|\mathbf{y}_{i,p} - \mathbf{y}_{i,q}\|_2 \right|. \tag{22}$$

All metrics are test-set averages in meters. Vertex and KNN edge-length metrics are weighted by vertices and KNN neighbor pairs, respectively, whereas centroid and triangle-edge errors weight objects equally. Both edge metrics are invariant to global rigid motion and are therefore diagnostic rather than ranking metrics for final placement.

The controlled terminal comparison evaluates all eight models at the exact 2 s target on the same 100 test scenes and 642-vertex sphere meshes. PIM and Small-PIM make one direct equilibrium-state

prediction, HOPNet performs 249 updates after consuming its required history, and the FIGNet and MeshGraphNet variants perform 250 updates from $t = 0$. Model-specific history, state representation, graph construction, and rollout conventions are retained.

HOPNet predicts rigid poses, FIGNet applies a rigid projection after each update, and MeshGraphNet uses a terminal-only rigid resolution bridge. Their high-resolution triangle-edge errors therefore reflect the output representation rather than unconstrained shape accuracy and are excluded from ranking. Vertex and centroid metrics are computed per object and then per scene before equal scene averaging. All rollout methods produced finite outputs on all 100 test scenes. Under the evaluator's general failure policy, failed or non-finite scenes are retained in the failure count and excluded only from error statistics explicitly conditioned on successful scenes. Paired scene-level confidence intervals use 10,000 bootstrap resamples.

D.2 Physical-Validity Metrics

For each rigid object, shape preservation is measured after optimal Kabsch alignment:

$$\varepsilon_{\mathrm{rigid}}(i) = \min_{\mathbf{R}\in \mathrm{SO}(3),\mathbf{t}\in\mathbb{R}^3} \sqrt{\frac{1}{n_i}\sum_{p} \|\mathbf{R}\mathbf{x}^0_{i,p} + \mathbf{t} - \widehat{\mathbf{y}}_{i,p}\|_2^2}. \tag{23}$$

This measures rigidity rather than final-pose accuracy.

Each mesh is sampled at vertices, unique-edge midpoints, and face centroids. For object pair (i, j), d_{ij}^{pen} is the maximum sampled penetration in both directions. A pair is valid when $d_{ij}^{\mathrm{pen}} \le 1\,\mathrm{mm}$; a scene is valid only when all pairs are valid. The supporting plane is not included.

For Flat, normalized analytic pairwise sphere overlap is

$$\varepsilon_{\mathrm{overlap}} = \frac{\sum_{i<j} V_{ij}^{\mathrm{overlap}}}{\sum_i V_i}. \tag{24}$$

For each object pair, ground-truth and predicted terminal contacts are determined from the simulated and predicted terminal meshes, respectively. A pair is classified as being in contact when its minimum triangle-surface distance is at most $1\,\mathrm{mm}$. Over all scene–object-pair instances, TP counts pairs classified as contacts in both the prediction and ground truth, FP counts predicted contacts absent from the ground truth, and FN counts ground-truth contacts missed by the prediction. These counts are micro-aggregated, and contact F1 is

$$F_1 = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}. \tag{25}$$

These metrics evaluate terminal inter-object geometry, not intermediate contact, forces, frictional feasibility, or support-plane penetration.

D.3 Runtime Protocol

PIM inference is measured on an RTX A6000 with PyTorch 2.4.0 in eager mixed precision, compilation disabled, 20 warm-up passes, and 100 measured passes. Device-resident timing includes only the model forward pass. MuJoCo 3.3.7 runs on two Intel Xeon Gold 6330 CPUs. Its measured per-scene runtime includes state initialization, keyframe restoration, rollout, and final vertex extraction, excluding XML and asset compilation.

Flat, Bumpy, and Jenga use three warm-up scenes and 100 test scenes; Deformable uses one warm-up and ten scenes. Single-scene latency pairs model batch size one with one MuJoCo worker. Throughput pairs model batch size B with B concurrent MuJoCo workers and excludes process startup, XML compilation, and dispatch.

The controlled Flat timing uses the first two manifest-ordered test scenes for latency and the first 16 for throughput. PIM and Small-PIM use batch size one for latency and batch size eight for throughput, whereas each rollout model uses one independent worker for latency and eight independent workers for throughput, each with batch size one. The matched $2\,\mathrm{s}$ equilibrium-state protocol measures one direct equilibrium-state prediction for PIM, 249 updates for HOPNet, and 250 updates for FIGNet

Table 9: Supplementary rollout timing for the six autoregressive models. HOPNet uses its released 102-update setting, while the FIGNet and MeshGraphNet reimplementations use a 100-update diagnostic. Latency uses two scenes with one worker, and throughput uses 16 scenes and eight workers.

Method	Calls	Latency (s)	Throughput (samples/s)
HOPNet	102	187.4733	0.021312
HOPNet-Adapt	102	168.3764	0.022649
FIGNet-Reimpl.	100	3.3253	0.645356
FIGNet-Reimpl.-Adapt	100	3.5275	0.637503
MeshGraphNet-Reimpl.	100	**1.4982**	**1.985109**
MeshGraphNet-Reimpl.-Adapt	100	1.6231	1.935324

and MeshGraphNet to reach 2 s. The supplementary 100/102-step rollout timing protocol excludes PIM and measures the released 102-update HOPNet rollout or a 100-update diagnostic for FIGNet and MeshGraphNet. Because the 102- and 100-update protocols are not identical, cross-family native-scale runtime ratios are descriptive rather than strictly task aligned.

Timing begins after checkpoint loading, scene loading, and static input preparation and ends after synchronized prediction. It includes required transfers, dynamic collision or interaction-graph construction, neural inference, autoregressive state updates, and rigid fitting where required, but excludes output serialization and error computation. Latency uses one worker and batch size one. Throughput uses batch size eight for PIM or eight concurrent rollout workers on the same RTX A6000. All models are measured sequentially with one OpenMP, MKL, OpenBLAS, and PyTorch CPU thread per process.

D.4 Supplementary 100/102-Step Rollout Timing

Table 9 preserves the model-native rollout scale as a runtime-only diagnostic; no accuracy result is reported at this horizon. PIM is excluded because it has no corresponding released or native-scale trajectory protocol. Within each model family, the variants without the additional force input and their corresponding *-Adapt* variants use identical call counts. HOPNet uses 102 updates, whereas FIGNet and MeshGraphNet use 100. Moreover, the original MeshGraphNets work does not define a rigid-body rollout protocol for this task. We therefore do not infer cross-family speedups from this table; the task-aligned 2 s terminal comparison remains the primary runtime result.

E Additional Results

E.1 Complete Quantitative Results

Table 10 reports the complete numerical comparison underlying Figure 4, including vertex, centroid, and edge errors. Table 12 provides the corresponding values for the controlled architectural ablations summarized in Figure 6 and the Deformable edge-loss ablation visualized in Figure 9. Table 13 reports the Deformable sensitivity study for the local neighborhood and source-anchor counts. Table 14 lists the complete accuracy and physical-validity metrics underlying the trade-off shown in Figure 15.

Relative to the full model, removing cross-object attention increases vertex error by 135.1%, 98.3%, 120.5%, and 87.6% on Flat, Bumpy, Jenga, and Deformable, respectively. The corresponding centroid-error increases are 180.6%, 109.2%, 202.4%, and 109.6%. The effect is therefore consistent across both rigid and deformable settings and is particularly pronounced for Jenga centroid prediction, where object placement depends on interactions throughout the stack. The lower edge errors obtained without cross-object attention on Flat, Bumpy, and Jenga do not contradict this conclusion: edge error measures local shape preservation and is insensitive to an incorrect global rigid pose.

The direct decoder produces higher vertex and centroid errors than the full dual-branch decoder in every scenario. Its vertex-error increases are 20.8%, 3.2%, 8.1%, and 1.3% on Flat, Bumpy, Jenga, and Deformable, respectively. Thus, explicitly separating object-level translation from zero-mean relative vertex motion reduces vertex and centroid errors in all four evaluated single-run comparisons

Table 10: Comparison with the direct equilibrium-state prediction baselines. All errors are in meters. Edge error measures local shape discrepancy relative to the ground-truth terminal mesh. Rigid-Pose MLP is evaluated on Deformable as a rigid-output reference, although it cannot represent non-rigid deformation. † denotes results from its rigid-transform output parameterization.

Scenario	Method	ε_{vtx}	ε_{cen}	$\varepsilon_{\text{edge}}$
Flat	PIM	**0.025695**	**0.020983**	1.182×10^{-5}
	Vertex Transformer	0.035674	0.033369	3.896×10^{-5}
	Rigid-Pose MLP	0.036876	0.035471	7.732×10^{-6} †
Bumpy	PIM	**0.027548**	**0.025536**	2.533×10^{-5}
	Vertex Transformer	0.033929	0.032083	3.455×10^{-5}
	Rigid-Pose MLP	0.035064	0.033051	7.690×10^{-6} †
Jenga	PIM	**0.020358**	**0.019961**	8.217×10^{-5}
	Vertex Transformer	0.029537	0.028590	5.751×10^{-5}
	Rigid-Pose MLP	0.029298	0.029221	1.071×10^{-5} †
Deformable	PIM	**0.019305**	**0.013211**	3.805×10^{-4}
	Vertex Transformer	0.044912	0.021323	3.680×10^{-4}
	Rigid-Pose MLP	0.032454	0.019386	7.889×10^{-4} †

Table 11: Jenga initial-condition and shape generalization without fine-tuning. ID-U uses the original input ranges without requiring impact or collapse; each initial-condition OOD set contains 90% matched-protocol and 10% outcome-unfiltered scenes. Shape OOD contains 2,000 matched-protocol scenes and changes only top-layer object J to an equal-volume cube. All errors are in meters and lower is better. † denotes edge errors from the rigid-transform output parameterization of Rigid-Pose MLP.

Condition	Method	ε_{vtx}	ε_{cen}	$\varepsilon_{\text{edge}}$
Matched ID	PIM	**0.020358**	**0.019961**	8.217×10^{-5}
	Vertex Transformer	0.029537	0.028590	5.751×10^{-5}
	Rigid-Pose MLP	0.029298	0.029221	$\mathbf{1.071 \times 10^{-5}}$ †
ID-unfiltered	PIM	**0.021130**	**0.020458**	8.203×10^{-5}
	Vertex Transformer	0.029082	0.027877	5.717×10^{-5}
	Rigid-Pose MLP	0.029089	0.029782	$\mathbf{1.070 \times 10^{-5}}$ †
Position OOD	PIM	**0.025523**	**0.023533**	8.793×10^{-5}
	Vertex Transformer	0.035871	0.035286	6.358×10^{-5}
	Rigid-Pose MLP	0.033835	0.034743	$\mathbf{1.070 \times 10^{-5}}$ †
Velocity OOD	PIM	**0.033473**	**0.031966**	9.128×10^{-5}
	Vertex Transformer	0.040956	0.040407	6.464×10^{-5}
	Rigid-Pose MLP	0.042400	0.044198	$\mathbf{1.068 \times 10^{-5}}$ †
Force OOD	PIM	**0.024587**	**0.023186**	8.501×10^{-5}
	Vertex Transformer	0.034687	0.032747	6.252×10^{-5}
	Rigid-Pose MLP	0.034308	0.034856	$\mathbf{1.071 \times 10^{-5}}$ †
Shape OOD	PIM	**0.036957**	**0.032656**	1.182×10^{-4}
	Vertex Transformer	0.041459	0.036174	3.737×10^{-4}
	Rigid-Pose MLP	0.045558	0.046572	$\mathbf{1.055 \times 10^{-5}}$ †

for terminal placement, but the small differences on Bumpy and Deformable indicate that the benefit is not uniformly large. On Jenga, the direct decoder reduces edge error by 29.8% while increasing vertex and centroid errors by 8.1% and 4.3%, again demonstrating that local geometric preservation and terminal placement evaluate different properties.

Table 12: Controlled single-run ablations evaluated on the current test splits. All errors are in meters and lower is better.

Scenario	Method	ε_{vtx}	ε_{cen}	$\varepsilon_{\text{edge}}$
Flat	Full	**0.025695**	**0.020983**	1.182×10^{-5}
	Direct decoder	0.031047	0.029016	4.739×10^{-5}
	No cross attention	0.060412	0.058884	1.181×10^{-5}
Bumpy	Full	**0.027548**	**0.025536**	2.533×10^{-5}
	Direct decoder	0.028434	0.026067	8.041×10^{-5}
	No cross attention	0.054619	0.053423	1.166×10^{-5}
Jenga	Full	**0.020358**	**0.019961**	8.217×10^{-5}
	Direct decoder	0.021999	0.020817	5.764×10^{-5}
	No cross attention	0.044889	0.060364	4.419×10^{-5}
Deformable	Full	**0.019305**	**0.013211**	3.805×10^{-4}
	Direct decoder	0.019547	0.013387	3.811×10^{-4}
	No cross attention	0.036225	0.027690	5.698×10^{-4}
	No edge loss	0.020123	0.013961	1.245×10^{-3}

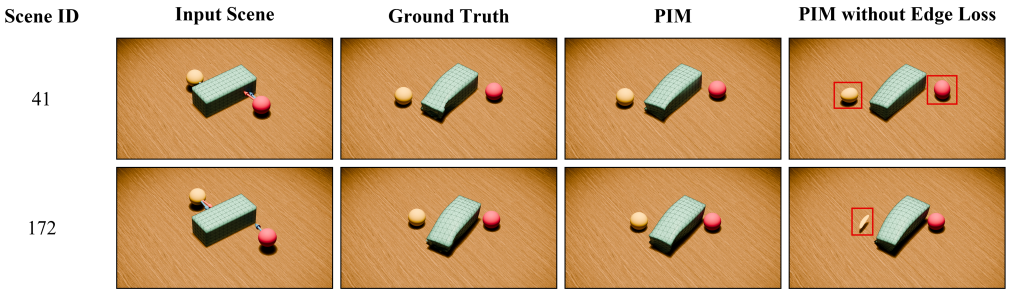


Figure 9: Qualitative effect of edge supervision on Deformable. Each row shows the input scene, ground-truth equilibrium state, full PIM prediction, and prediction from PIM trained without the edge loss. Red boxes highlight local shape distortions in the predictions without edge supervision. Aggregate results over all 200 test scenes are reported in Table 12.

Removing the edge loss on Deformable increases vertex and centroid errors by 4.2% and 5.7%, respectively, but increases the KNN edge-length error by 227.2%. The edge error therefore detects a loss of local geometric fidelity that is weakly reflected by aggregate pointwise placement metrics. Figure 9 shows the corresponding visible distortions, particularly on the nominally rigid spheres, illustrating that KNN edge-length supervision helps preserve local object geometry while still allowing the soft box to deform toward the ground-truth terminal geometry.

At fixed $a = 32$, reducing the neighborhood to $k = 16$ lowers fine-stage step time by 21.7% while increasing vertex error by 1.18%. Increasing the neighborhood to $k = 64$ raises step time by 45.4%, reduces vertex error by only 0.23%, and increases edge error by 1.55%. At fixed $k = 32$, reducing the anchor count to $a = 16$ increases vertex and centroid errors by 0.23% and 0.18%, respectively, while reducing edge error by 0.71%. Increasing the count to $a = 64$ increases vertex and centroid errors by 0.50% and 0.54%, respectively, while reducing edge error by 0.21%. The measured step times for both alternatives remain within 1.3% of the default. Thus, $a = 32$ provides the best terminal-placement accuracy in this sweep, whereas fewer anchors provide only a small edge-preservation improvement.

Table 13: Deformable hyperparameter sensitivity in matched single-run experiments using seed 1. Errors are test-set averages in meters and lower is better. Relative time is normalized by the matched $(k, a) = (32, 32)$ reference. The edge neighborhood is fixed at 32, and all variants have the same number of learnable parameters.

Variant	(k, a)	ε_{vtx}	ε_{cen}	$\varepsilon_{\text{edge}}$	Relative step time
Smaller KNN	$(16, 32)$	0.019769	0.013637	3.825×10^{-4}	0.783
Default (sweep)	$(32, 32)$	0.019539	0.013430	3.807×10^{-4}	1.000
Larger KNN	$(64, 32)$	0.019494	0.013441	3.866×10^{-4}	1.454
Fewer anchors	$(32, 16)$	0.019583	0.013454	3.780×10^{-4}	0.987
More anchors	$(32, 64)$	0.019636	0.013503	3.799×10^{-4}	0.988

Table 14: Accuracy–validity trade-off of the Flat-specific non-penetration extension. Mean maximum penetration is the test-set average of the sampled per-scene maximum penetration depth. Arrows indicate preferred direction.

Variant	$\varepsilon_{\text{vtx}} \downarrow$	$\varepsilon_{\text{cen}} \downarrow$	Mean max pen. (mm) $\downarrow$	Valid pairs (%) $\uparrow$	Overlap (%) $\downarrow$	Valid scenes (%) $\uparrow$	Contact F_1 (%) $\uparrow$
PIM (main)	0.025695	0.020983	13.967	94.83	0.668	10.90	13.16
$\lambda_{\text{pen}} = 0.1$	0.025834	0.021111	8.890	96.03	0.290	20.90	13.02
$\lambda_{\text{pen}} = 0.3$	0.026402	0.021707	5.552	97.17	0.136	34.30	12.05
$\lambda_{\text{pen}} = 1.0$	0.027633	0.023166	2.900	98.31	0.051	53.60	11.11

E.2 Additional Qualitative Results

Figure 10 shows one held-out controlled Flat scene comparing direct equilibrium-state prediction with autoregressive rollout. The remaining Figures 11–14 provide four additional PIM test examples for each scenario. For each scenario, four test-set indices were sampled uniformly without replacement using a fixed scenario-specific random seed. The selection did not use prediction errors or physical-validity outcomes. In these scenario-specific figures, rows correspond to sampled test scenes and columns show the initial scene, MuJoCo final state, PIM prediction, and per-vertex error, following the layout of Figure 3. Red arrows indicate initial external forces, and blue arrows indicate initial velocities.

F Physics-Aware Extension Details

The main PIM objective contains only supervised geometric losses and does not explicitly penalize inter-object penetration. As a scenario-specific extension on Flat, we evaluate a differentiable non-penetration regularizer. This setting admits an analytic proxy because every object is a sphere with nominal radius $r = 0.05\,\text{m}$ and the supporting plane has height $h = 0$. The regularizer penalizes analytic sphere–sphere and sphere–plane penetration proxies during fine-stage training while leaving the architecture and inference procedure unchanged. Because the proxy relies on nominal sphere geometry and a known plane, we treat it as a Flat-specific extension rather than a general collision-validity solution.

For sample b, let $\widehat{\mathbf{c}}_{b,i}$ be the predicted vertex mean. The sphere–sphere and sphere–plane penetration proxies are

$$\delta^{\text{obj}}_{b,ij} = \left[2r - \|\widehat{\mathbf{c}}_{b,i} - \widehat{\mathbf{c}}_{b,j}\|_2 - \tau_{\text{train}}\right]_+ , \tag{26}$$

$$\delta^{\text{surf}}_{b,i} = \left[h + r - \widehat{c}_{b,i,z} - \tau_{\text{train}}\right]_+ . \tag{27}$$

The normalized regularizer is

$$\mathcal{L}_{\text{pen}} = \frac{1}{B\binom{M}{2}} \sum_b \sum_{i<j} \left(\frac{\delta^{\text{obj}}_{b,ij}}{s}\right)^2 + \frac{1}{BM} \sum_b \sum_i \left(\frac{\delta^{\text{surf}}_{b,i}}{s}\right)^2 , \tag{28}$$

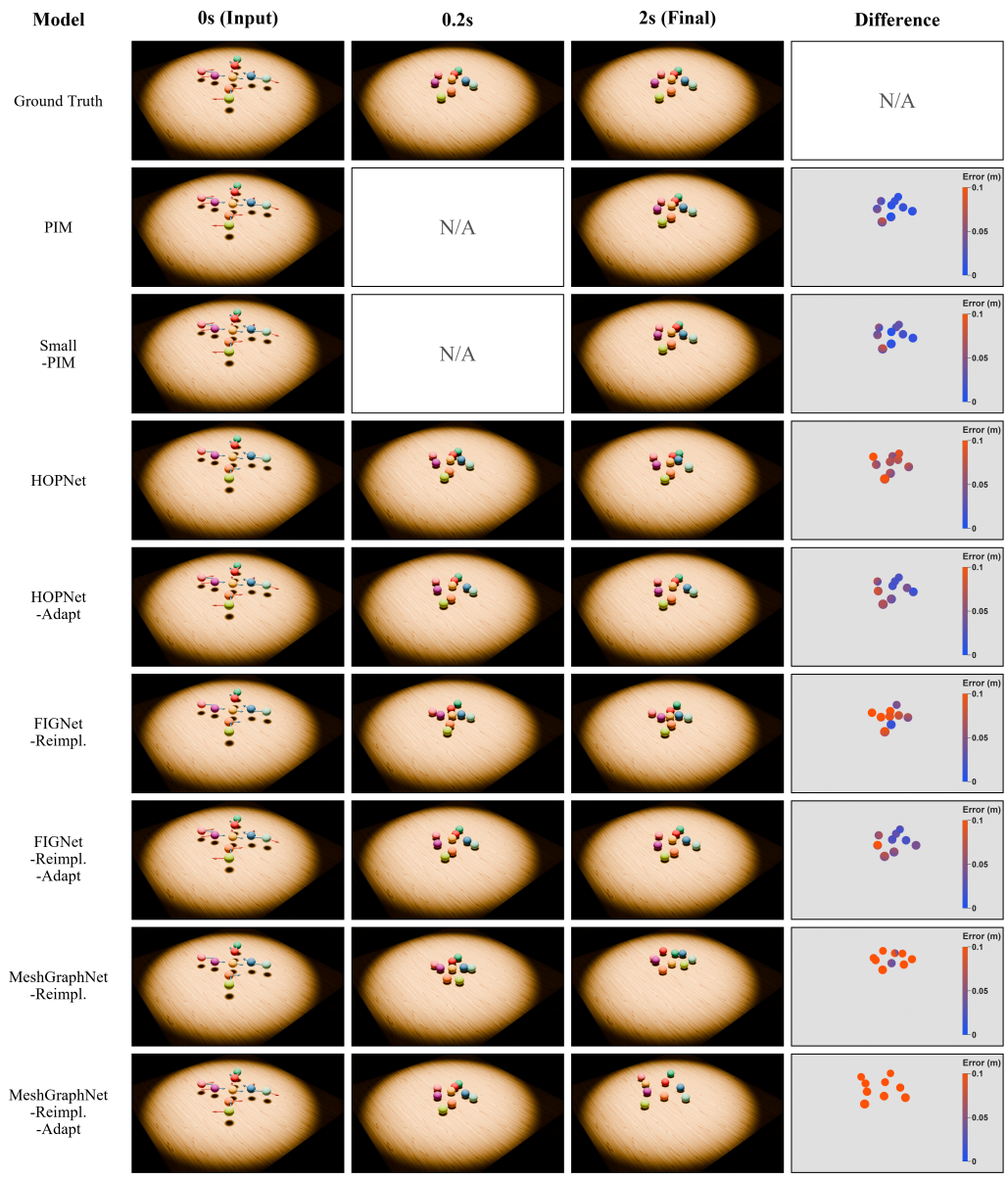


Figure 10: Qualitative comparison on one held-out scene from the controlled Flat experiment. Rows show the ground-truth trajectory, PIM, Small-PIM, and the six autoregressive rollout baselines. Columns show the input at $t = 0$, the intermediate state at $t = 0.2\,\mathrm{s}$, the equilibrium state at $t = 2\,\mathrm{s}$, and the corresponding object-wise terminal errors. PIM and Small-PIM predict the equilibrium state directly and therefore do not produce an intermediate state. All error visualizations use the same 0–$0.1\,\mathrm{m}$ color scale. Reimpl. denotes a PyTorch reimplementation developed for this comparison.

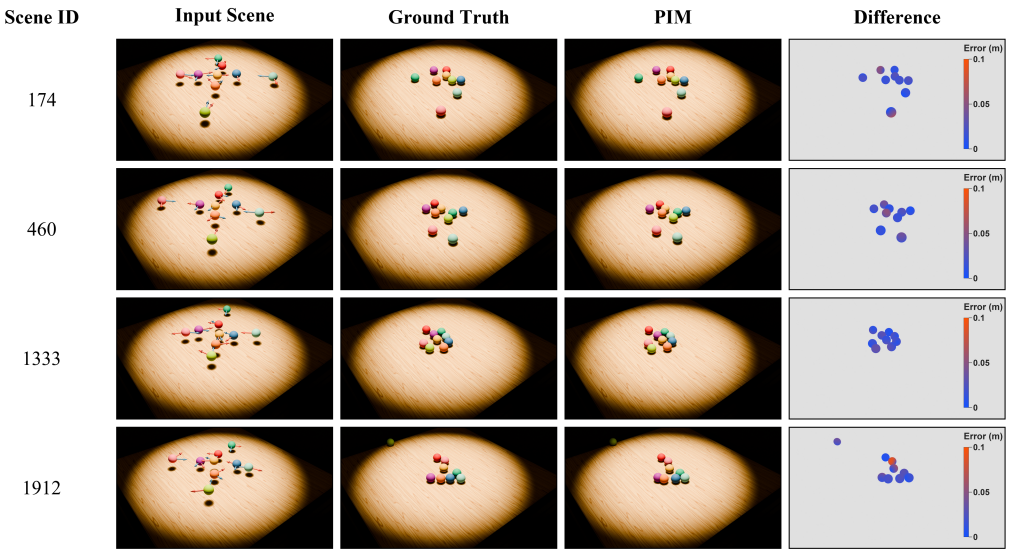


Figure 11: Additional randomly sampled qualitative results on Flat. Each row shows one held-out test scene.

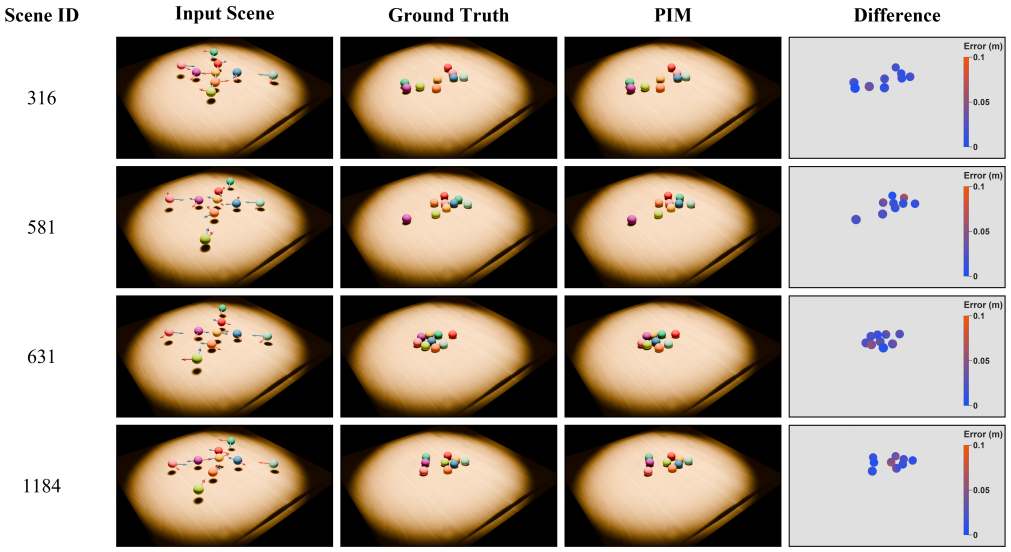


Figure 12: Additional randomly sampled qualitative results on Bumpy. Each row shows one held-out test scene.

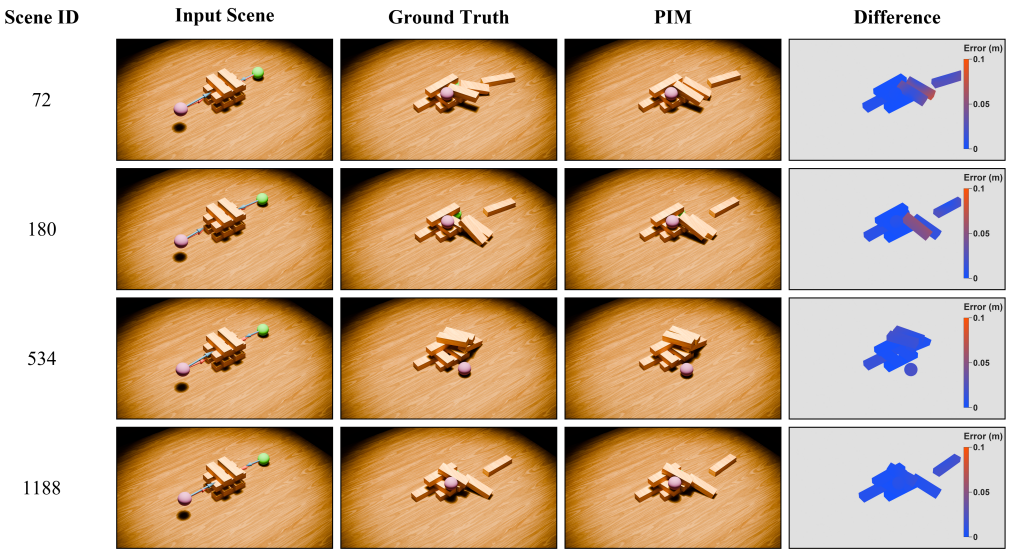


Figure 13: Additional randomly sampled qualitative results on Jenga. Each row shows one held-out test scene.

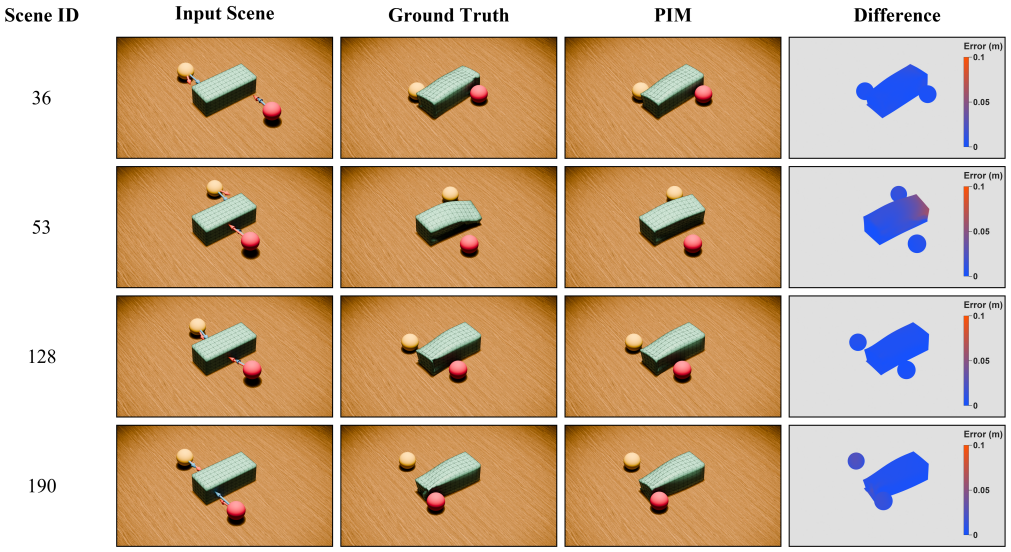


Figure 14: Additional randomly sampled qualitative results on Deformable. Each row shows one held-out test scene.

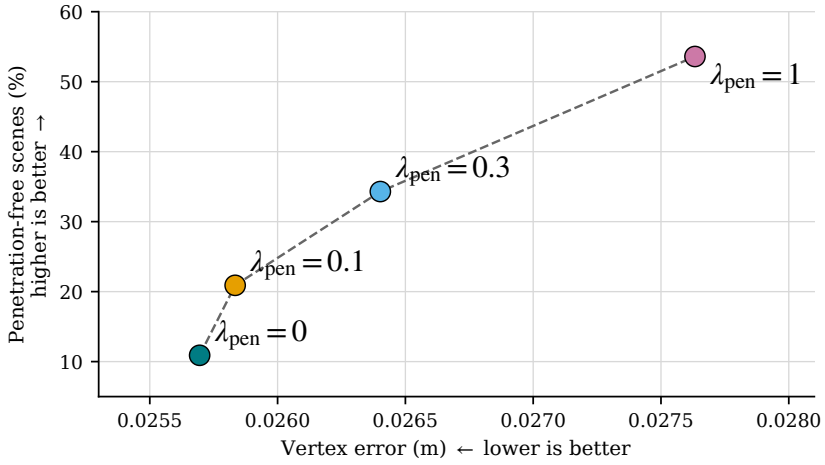


Figure 15: Accuracy–validity trade-off of the Flat-specific non-penetration extension. The horizontal axis shows mean vertex error, while the vertical axis shows the penetration-free scene rate. Lower vertex error and higher penetration-free rate are preferred. Exact metric values are provided in Table 14.

where $s = 0.05\,\mathrm{m}$ and $\tau_{\text{train}} = 0.5\,\mathrm{mm}$. Fine training minimizes $\mathcal{L}_{\text{ext}} = \mathcal{L} + \lambda_{\text{pen}}\mathcal{L}_{\text{pen}}$.

All three variants initialize from the original Flat coarse checkpoint and apply the regularizer only during 40 fine-stage epochs. Optimizer, splits, reconstruction losses, and checkpoint selection are unchanged. We test $\lambda_{\text{pen}} \in \{0.1, 0.3, 1.0\}$. The regularizer has $O(BM^2)$ training complexity, is used only during training, and does not alter the inference procedure.

Results. Figure 15 shows the accuracy–validity trade-off. With $\lambda_{\text{pen}} = 0.1$, the penetration-free scene rate increases from 10.90% to 20.90%, while the mean sampled maximum penetration decreases from 13.967 to $8.890\,\mathrm{mm}$ and vertex error increases by only 0.54%. Increasing λ_{pen} to 0.3 and 1.0 further raises the penetration-free rate to 34.30% and 53.60%, with vertex-error increases of 2.75% and 7.54%, respectively. Contact F_1 decreases from 13.16% to 13.02%, 12.05%, and 11.11% across the three weights, showing that suppressing penetration does not by itself recover the ground-truth terminal contact graph. We retain the original PIM as the main model because the analytic proxy is specific to the Flat sphere–plane setting, and treat the regularized variants as optional scene-specific extensions. Table 14 provides all metrics.