Supplementary Information: Integrating Physics and Topology in Neural Networks for Learning Rigid Body Dynamics

Amaury Wei
IMOS Laboratory
EPFL, Switzerland
amaury.wei@epfl.ch

Olga Fink*
IMOS Laboratory
EPFL, Switzerland
olga.fink@epfl.ch

*To whom correspondence should be addressed.

1 Supplementary Notes

Supplementary Table 1: Important Symbols and Notations

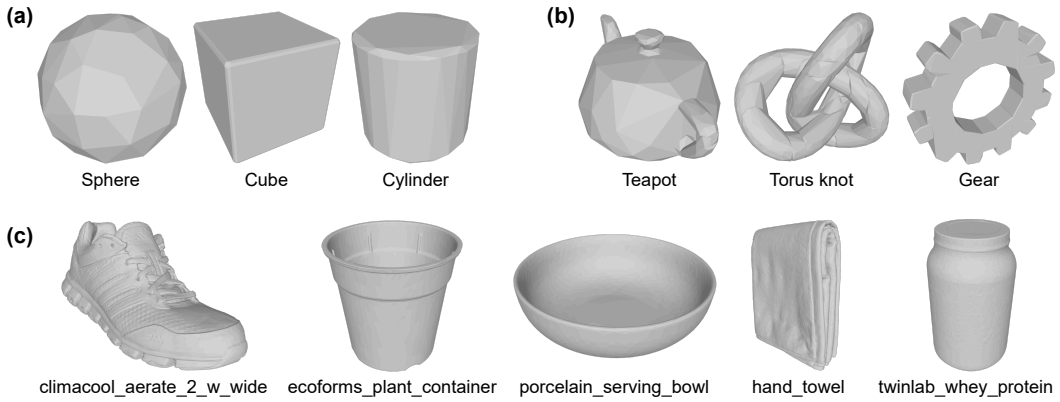
Notation	Meaning
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Graph representation of an object mesh with vertices $\mathcal{V}$ and edges $\mathcal{E}$
$\mathcal{V} = \{v_i\}_{i=1..N}$	Set of N vertices in a graph $\mathcal{G}$ corresponding to mesh nodes
$\mathcal{E} = \{e_{v_s \to v_r} \vert v_s \neq v_r\}$	Set of edges in a graph $\mathcal{G}$ between a sender node v_s and receiver node v_r
$\mathcal{X}^{t_0}$	Combinatorial Complex (CC) representation of a mesh at time t_0
$\hat{\mathcal{X}}^{t_1}$	Estimated Combinatorial Complex (CC) representation of a mesh at time t_1
r	Rank of a cell in a CC defining its hierarchy
$X^{(r)}$	Cell of rank r in a CC
$\boldsymbol{h}^{(r)}$	Features of a cell $X^{(r)}$ of rank r in a CC
$\boldsymbol{x}^t \in \mathbb{R}^3$	Ground-truth position of an object or node at time t
$\hat{\boldsymbol{x}}^t \in \mathbb{R}^3$	Estimated position of an object or node at time t
$\dot{\boldsymbol{x}}^t \in \mathbb{R}^3$	Velocity of an object or node at time t
$\ddot{\boldsymbol{x}}^t \in \mathbb{R}^3$	Ground-truth acceleration of an object or node at time t
$\hat{\ddot{\boldsymbol{x}}}^t \in \mathbb{R}^3$	Estimated acceleration of an object or node at time t
$\boldsymbol{q}^t \in \mathbb{R}^4$	Ground-truth quaternion orientation of an object at time t
$\hat{\boldsymbol{q}}^t \in \mathbb{R}^4$	Estimated quaternion orientation of an object at time t
$\boldsymbol{n}^t \in \mathbb{R}^3$	Normal vector of a mesh triangle $X^{(2)}$ at time t
$\boldsymbol{p}_s^t \in \mathbb{R}^3$	Closest point on a mesh triangle $X_s^{(2)}$ between two mesh triangles $X_s^{(2)}$ and $X_r^{(2)}$
$\{\boldsymbol{x}_{s_j} \in \mathbb{R}^3\}_{j=1,2,3}$	Positions $\boldsymbol{x}$ of the three nodes belonging to a mesh triangle $X_s^{(2)}$
d_c	Collision radius threshold under which a collision contact cell $X^{(3)}$ is created
$\text{rk}(x)$	Rank function returning the rank r of a cell $x \in \mathcal{X}$

2 Supplementary Methods

2.1 Dataset Details

The MOVi datasets used for the experiments have been generated with the Kubric library [1]. They consist of rigid-body simulations of multiple 3D objects tossed simultaneously onto a floor. The simulations are computed with the Bullet [2] simulator. The datasets (MOVi-spheres, MOVi-A, MOVi-B) have different increasing levels of complexity, defined by the shape of the objects.

Each experiment contains 3 to 10 objects tossed toward a target location on the floor. MOVi-A uses simple shapes with coarse meshes: cubes (51 nodes), spheres (64 nodes), and cylinders (64 nodes). MOVi-B uses more complex shapes of everyday objects with meshes up to 1590 nodes. Finally, MOVi-C further increases complexity by using objects from the Google Scanned Objects dataset [3], which consists of high-resolution 3D scans of real-world household items such as shoes, hats, bags, toys, and bottles. These objects have significantly finer meshes, averaging 7917 nodes per object, with some reaching up to 50561 nodes.



Supplementary Figure 1: Example of objects in the MOVi datasets. (**a**) All MOVi-A objects; (**b**) Some MOVi-B objects; (**c**) Some MOVi-C objects. MOVi-B and -C objects have been scaled to fit.

The size of each object in the MOVi datasets is determined based on its category. In MOVi-A and MOVi-B, objects are initially defined with a mean dimension of 1 [m] (e.g., spheres with a 1 [m] diameter, cubes with 1 [m] sides, ...). Each object is then randomly scaled to either a small (0.7) or large (1.4) size category, resulting in final dimensions of $(0.7, 1.4)$ [m] for all object types. In contrast, objects in MOVi-C retain their real-world sizes, as they are derived from high-resolution 3D scans. For example, shoes are approximately 0.27 [m] in length, while tables measure around 1.5 [m]. Additionally, each object in MOVi-C is scaled by a randomly sampled factor within the range $[0.75, 3.0]$, introducing further variability in object sizes across simulations.

Each object is also randomly assigned a material type (metal or rubber). Each material has three physical properties (friction, restitution, density) with values of $(0.4, 0.3, 2.7)$ and $(0.8, 0.7, 1.1)$ for metal and rubber respectively. The initial conditions (position, orientation, velocity) of each individual object are also determined randomly for each experiment.

In addition to the physical state of each object, Kubric also renders images for vision-based tasks using Blender [4]. However, in this work, we only use the object position and rotation quaternion. Rendered images are used for visualization purposes only.

All datasets have been generated with the same parameters, given in Table 2. Each dataset contains 1200 individual experiments in total, each one generated with a unique seed value in $[1 - 1200]$.

Supplementary Table 2: Kubric parameters used to generate the MOVi datasets.

Simulation Rate	Output Data Rate	Experiment Duration	Image Resolution
1200 [Hz]	240 [Hz]	2 [sec]	256 x 256 [pixels]

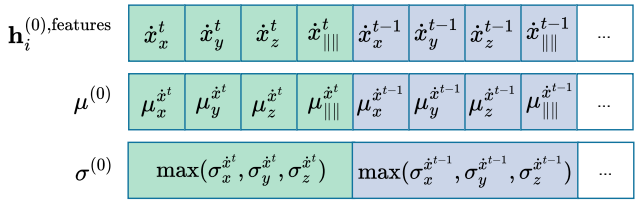
Each dataset is randomly divided into 80% training, 10% validation, and 10% testing samples. The same dataset split is consistently used across all experiments, ensuring comparability between in-domain and generalization tasks.

2.2 Model Details

Architecture The proposed topological neural network uses an Encode-Process-Decode architecture [5]. All Multi-Layer Perceptrons (MLPs) in our models are ReLU-activated two-hidden-layer MLPs with hidden and output size of 128 (except the decoder MLPs with output size 3). All MLPs, except for the decoders, are followed by a LayerNorm [6] layer. In total, each standard model has $707\,708$ learnable parameters.

Message passing A detailed overview of the message-passing strategy is given in Supplementary Figure 3. The equation numbers match the main text.

Normalization All input and target features are normalized to have zero mean and unit variance, using dataset-specific statistics. The mean vector $\boldsymbol{\mu}$ is computed component-wise, while the scaling vector $\boldsymbol{\sigma}$ is adjusted with 3D-equivariance in mind. Since most features are geometric (e.g., 3D velocities, 3D distances), the model must be equivariant [7] to object orientation. For example, if momentum is transferred between the Z-axis and X-axis during a collision, applying different scalings to these axes could violate energy conservation and degrade model accuracy. An example of this normalization strategy applied to node features is illustrated in Supplementary Figure 2.



Supplementary Figure 2: Normalization strategy applied to the node features $\boldsymbol{h}^{(0),\textbf{features}}$. The mean vector $\boldsymbol{\mu}^{(0)}$ contains different coefficients for each dimension, while the scaling vector $\boldsymbol{\sigma}^{(0)}$ contains a unique scaling factor applied to all components of a 3D+norm vector.

The formula used to normalize features for a cell $x_i^{(r)}$ of rank r is given in Supplementary Equation 1:

$$\boldsymbol{h}_{i,\text{normalized}}^{(r),\text{features}} = \frac{\boldsymbol{h}_i^{(r),\text{features}} - \boldsymbol{\mu}^{(r)}}{\boldsymbol{\sigma}^{(r)}} \tag{1}$$

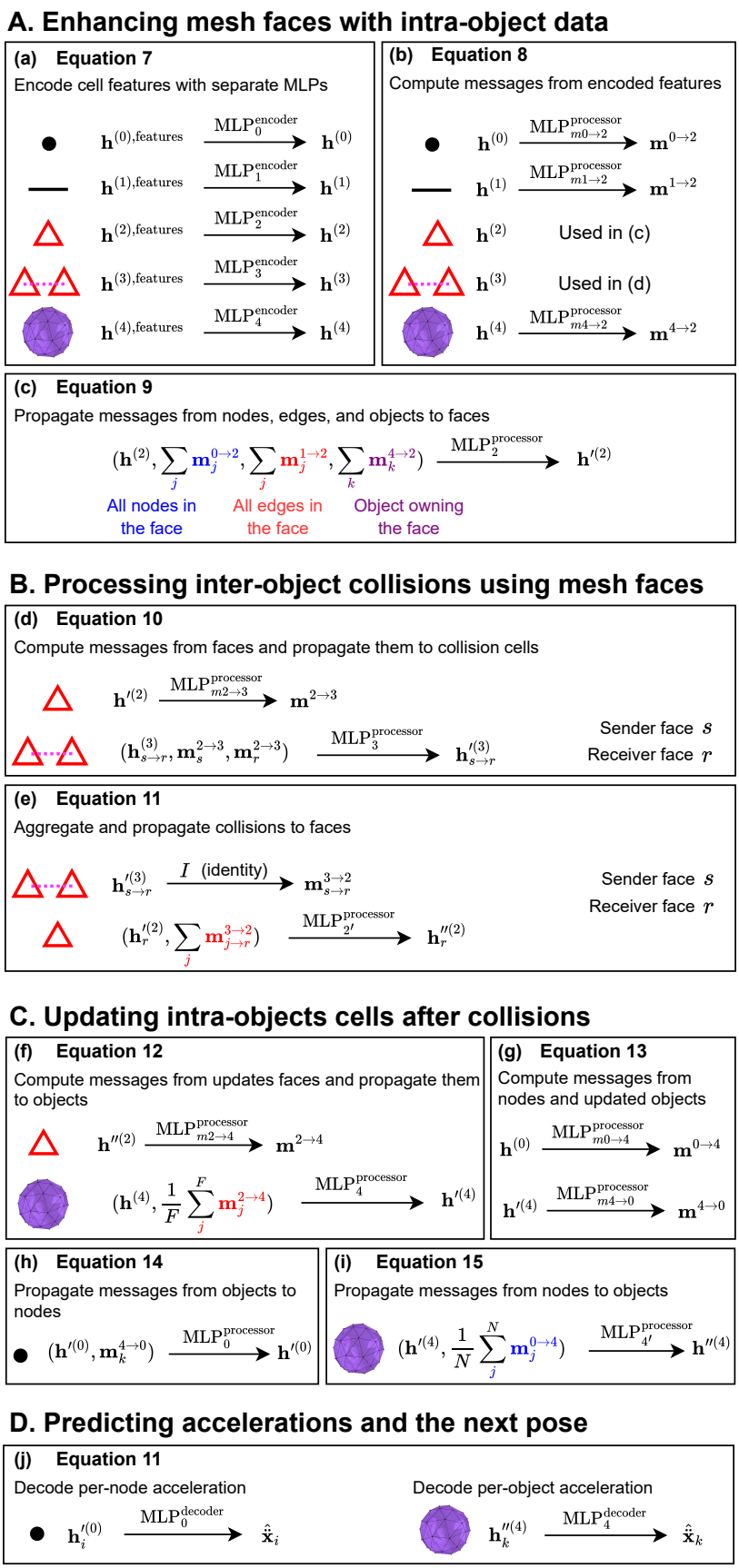
Loss The training target at each timestep t is the per-node (rank 0) and per-object (rank 4) acceleration $\ddot{\boldsymbol{x}}_i^t$. The loss is computed as the Mean-Squared-Error (MSE) on the predicted accelerations in the normalized target space, making the entire framework invariant to the scale of the data.

2.3 Training Details

Optimization Each model is trained separately on its respective dataset using the Adam optimizer [8]. The learning rate schedule is selected individually for each model, employing either a constant learning rate of 10^{-4} or an exponential decay over 40 epochs ($\gamma = 0.9441$). The choice of schedule is determined based on the model's best rollout performance after 40 epochs. Exact details for each model are given in Supplementary Table 3.

Batching During training, each experiment is processed one by one. The order of experiments is randomly shuffled at the beginning of each epoch. The model is trained to predict the acceleration $\ddot{\boldsymbol{x}}_i^t$ for each node and object at timestep $t \in [2 - 480]$. Gradients are computed for each timestep and aggregated for the whole experiment. The model weights are updated once per experiment, resulting in batches of 478 timesteps.

Hardware Models were trained on a single RTX 3090 GPU paired with an i7-9700k and 32GB of RAM. However, with the models being fairly small, GPUs with smaller amounts of VRAM are sufficient.



Supplementary Figure 3: Detailed overview of the HOPNet message passing.

Supplementary Table 3: Learning rate schedules used for each model.

Dataset	Model Features	Constant Rate	Exponential Decay
MOVi-spheres	All features		10^{-4} to 10^{-5}
MOVi-A	All features		10^{-4} to 10^{-5}
MOVi-B	All features		10^{-4} to 10^{-5}
MOVi-A	3-layers MLPs	10^{-4}	
MOVi-A	Two MP layers		10^{-4} to 10^{-5}
MOVi-A	64-embeddings		10^{-4} to 10^{-5}
MOVi-A	No object cells $X^{(4)}$		10^{-4} to 10^{-5}
MOVi-spheres	No center-mass distance	10^{-4}	
MOVi-A	No center-mass distance		10^{-4} to 10^{-5}
MOVi-A	Non-sequential MP		10^{-4} to 10^{-5}

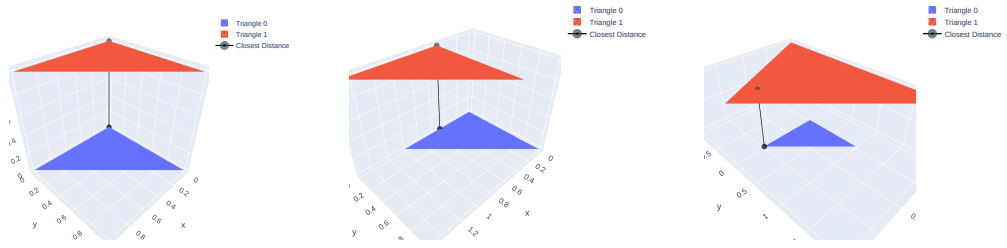
2.4 Special collisions between parallel mesh triangles

In extremely rare cases, collisions could occur between parallel mesh triangles and lead to multiple closest points. This section clarifies how HOPNet determines the closest points and how the feature embedding is computed in such cases.

While multiple closest points can theoretically exist when two triangles are perfectly parallel, HOPNet always selects a single closest point per triangle ($\boldsymbol{p}_s$ and $\boldsymbol{p}_r$) for feature embedding (see Equation (5) in the main article).To illustrate how this point is determined in parallel cases, Supplementary Figure 4 presents three scenarios: (1) fully overlapping triangles where the closest points are corners (Suppl. Fig. 4a); (2) laterally shifted identical triangles where one closest point remains the same while the other moves to an edge (Suppl. Fig. 4b); and (3) non-identical parallel triangles where one closest point is a vertex, and the other is its projection (Suppl. Fig. 4c).

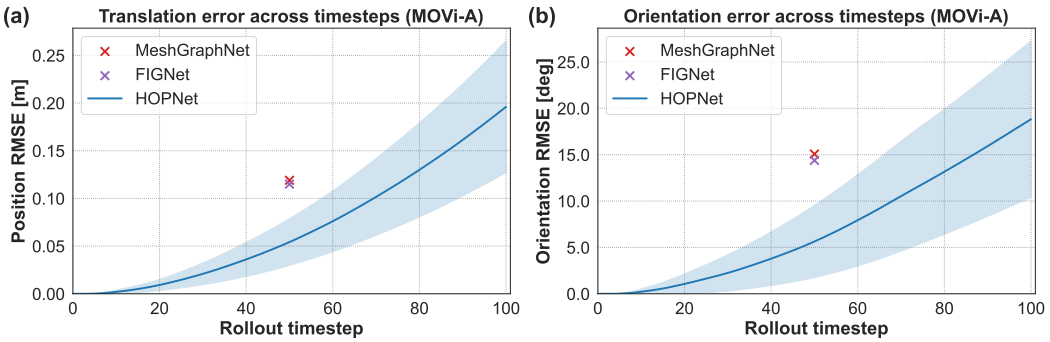
Since the feature embedding for each collision includes the displacement vectors and norms $[\boldsymbol{x}^t_{s_i} - \boldsymbol{p}^t_s, \|\boldsymbol{x}^t_{s_i} - \boldsymbol{p}^t_s\|]_{i=1,2,3}$ and $[\boldsymbol{x}^t_{r_i} - \boldsymbol{p}^t_r, \|\boldsymbol{x}^t_{r_i} - \boldsymbol{p}^t_r\|]_{i=1,2,3}$ for each triangle s and r, special cases occur when a closest point coincides with a corner. In these cases, the displacement vector is a zero vector, which leads to zero-valued entries in the embedding. In Supplementary Figure 4a, both closest points are at the corners, leading to four zero-vectors in the feature embedding. In Supplementary Figure 4b, one closest point remains a corner while the other moves along an edge, resulting in two zero-vectors. In Supplementary Figure 4c, the closest point on one triangle is a vertex, and the other is its projection, leading to two zero-vectors.

However, such cases are extremely rare in practice, as mesh surfaces are almost never perfectly parallel due to numerical precision limits and object motions. Consequently, the closest points are almost always offset from triangle edges rather than aligning exactly at mathematically perfect locations.



(a) Vertical shift **(b)** Vertical + lateral shift **(c)** Different triangles

Supplementary Figure 4: Different scenarios of parallel triangles and their computed closest points. (**a**) Identical triangles with a vertical offset: the closest points are corresponding corners of both triangles. (**b**) Identical triangles with both vertical and lateral offsets: the closest point of triangle 1 remains a corner, while its projection falls on an edge of triangle 0. (**c**) Non-identical parallel triangles: the closest point of triangle 0 is a corner, while its projection lies within triangle 1.



Supplementary Figure 5: Evolution of the autoregressive rollout errors. (**a**) Translation and (**b**) orientation errors over time. The error band indicates the standard deviation over the complete test set using the median seed. Performance comparisons with FIGNet [9] and MeshGraphNet [10] are based on the results reported in [9].

3 Supplementary Discussion

3.1 Performance on long rollouts

The main performance evaluation of our framework is presented in Section 3.1 of the main text with a rollout horizon of $T = 50$, aligning with the baseline results reported for FIGNet [9] and MeshGraphNet [10] in [9], as reproducible code for these models is unavailable. However, assessing our method's performance on extended autoregressive rollouts is essential for evaluating its stability and accuracy over long simulation times.

Beyond the standard 50-step horizon, we also evaluate our framework's performance on both short- and long-term predictions, considering different rollout horizons T ranging from 25 to 100 steps. Supplementary Figure 5 illustrates this analysis on the `MOVi-A` dataset.

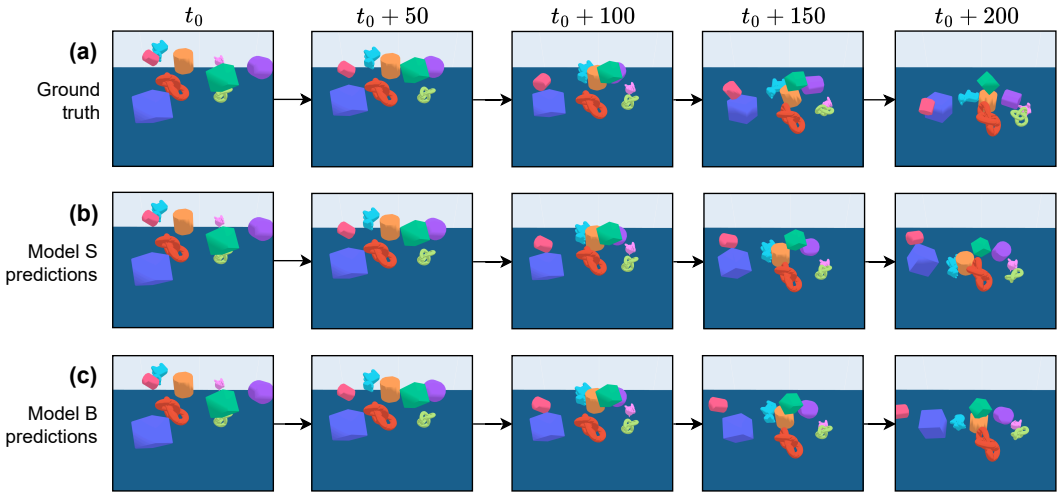
As shown in Supplementary Figure 5, the translation and orientation errors increase exponentially over time, an expected trend due to the cumulative errors at each timestep. The coarse mesh resolution in `MOVi-A` magnifies small errors in position and orientation, leading to larger deviations in collision outcomes and contributing to the increasing standard deviation over time. Importantly, this error growth stems from subtle variations in initial collision conditions rather than any inherent limitation in our model's ability to capture the underlying dynamics accurately.

Compared to the best baseline model, our method can perform 50% more rollout steps before reaching the same translational and rotational errors. On the `MOVi-A` dataset, FIGNet [9] reaches an RMSE of 0.115 [m] at 50 steps, while our method reaches a comparable error of 0.117 [m] at 75 steps. This significant improvement allows more accurate predictions further into the future, supporting better and more robust decision-making in dynamic scenarios.

For clarity and ease of comparison, we also present these extended results in tabular format. Supplementary Tables 4 and 5 summarize the performance comparisons with baseline models on the `MOVi-A` and `MOVi-B` datasets, respectively.

3.2 Generalization rollout

To demonstrate HOPNet's ability to generalize to unseen complex object geometries, we trained three models (S, A, and B) on the `MOVi-spheres`, `MOVi-A`, and `MOVi-B` datasets. After evaluating model S on the `MOVi-A` and `MOVi-B` datasets, which feature significantly more complex object geometries, HOPNet achieves performance comparable to—or better than—models A and B. This generalization capability is illustrated in Supplementary Figure 6 where we test both models S and B on video n°1 of the `MOVi-B` dataset, and compare the generated rollouts against the ground truth.

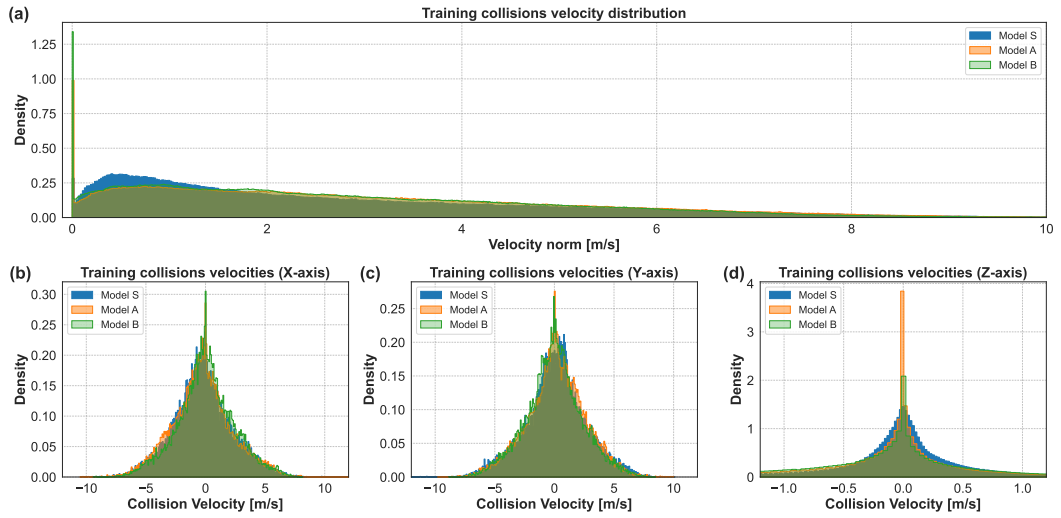


Supplementary Figure 6: Generalization rollout example. (**a**) Ground truth; (**b**) Rollout generated by model S, trained solely on spherical objects with `MOVi-spheres`; (**c**) Rollout generated by model B, trained on `MOVi-B`. The objects in this scenario have never been encountered by model S which still manages to generate accurate and realistic rollouts. This sequence is taken from experiment 10 of the `MOVi-B` dataset.

3.3 Collision velocity bias and its impact on generalization

As discussed in Section 3.2 of the main article, model S—despite being trained only on spherical objects—generalizes better to complex geometries than model A, and even has a slightly lower orientation error than model B on `MOVi-B`. We hypothesize that this counterintuitive result is due to differences in the training data: spheres in the `MOVi-spheres` dataset roll continuously, leading to a more evenly distributed range of collision velocities. In contrast, the objects in `MOVi-A` and `MOVi-B` have flat surfaces and sharp edges, which often cause them to slow down or come to a stop, resulting in a bias toward static or low-speed contacts.

Supplementary Figure 7 shows the distribution of collision velocities in the training datasets. The `MOVi-spheres` dataset (used for model S) features a Gaussian-like distribution of collision speeds across all axes. In contrast, `MOVi-A` and `MOVi-B` (used for models A and B) are dominated by near-zero Z-axis velocities, reflecting a bias towards static contacts due to the nature of the object geometries.

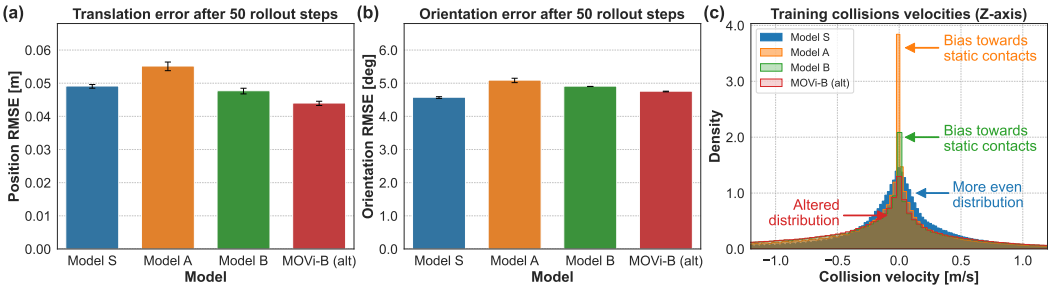


Supplementary Figure 7: Distribution of the training collision velocities per model. (**a**) Velocity norm; (**b**) X-axis component; (**c**) Y-axis component; (d) Z-axis component. Models A and B are trained on mostly static and slow collisions, while model S is trained on a more diverse set. The X- and Y-axis velocities have a similar distribution between all datasets. However, the Z-axis velocities are closely distributed around 0 for A and B, indicating mostly static and slow-speed vertical collisions.

To validate our hypothesis, we trained a new model *MOVi-B (alt)* using a modified training strategy: we masked the loss for colliding objects with Z-axis velocities below 10^{-3} [m/s]. This reduces the model's bias toward static contacts while preserving the rest of the velocity distribution (Suppl. Fig. 8c).

This modified *MOVi-B (alt)* yields improved performance on the `MOVi-B` dataset, with a translation error of 44.019 [mm] versus 47.725 [mm] for model B (a 7.76% reduction), and an orientation error of 4.759 [deg] versus 4.911 [deg] (a 3.09% reduction) (Fig. 8a,b). Its orientation error remains 0.182 [deg] higher than model S, which is expected due to the richer angular collision diversity in `MOVi-spheres`. However, it significantly outperforms model S in translation, likely because center-of-mass distances are more diverse in `MOVi-B`, where objects vary in shape, unlike spheres which have uniform geometry.

Overall, these results validate our hypothesis: reducing the bias toward low-velocity contacts improves generalization by exposing the model to a more uniform range of collision dynamics. These findings highlight a key strength of our approach: the ability to generalize beyond the training distribution to new complex object geometries while maintaining accurate predictions. This is particularly valuable for counterfactual reasoning, which requires reliable long-horizon rollouts under new environment configurations.



Supplementary Figure 8: Performance of different models on the `MOVi-B` dataset. (**a**) Translation and (**b**) orientation errors after a rollout horizon of $T = 50$ on all dynamic objects in the scene. The *MOVi-B (alt)* model has been trained on an altered version of `MOVi-B` dataset, where training stops at timestep $t = 400$. Error bars indicate the mean and standard deviation across three independent random seeds.

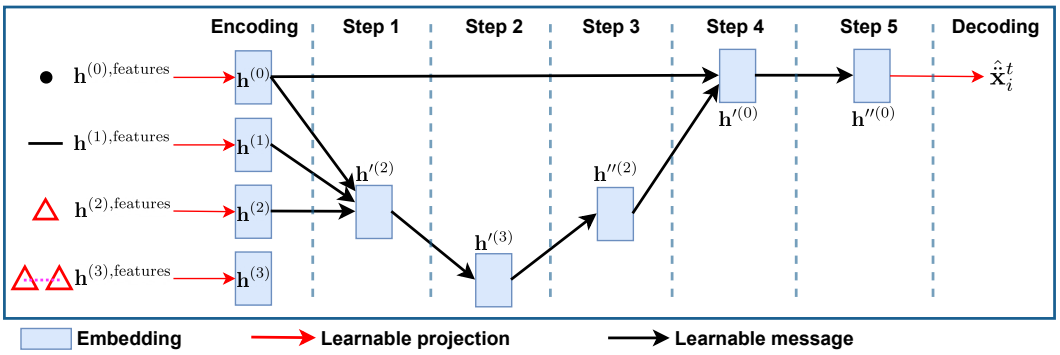
3.4 Model details used for ablation

Ablation on topological representation To highlight the significance of our proposed topological representation $\mathcal{X}$, we conduct experiments by removing the higher-order topological groups representing individual objects, specifically the $X^{(4)}$ cells. To maintain a fair comparison, we transfer the constant features of the $X^{(4)}$ cells features to the node features. Therefore, each node v_i inherits the object type, mass, friction, and restitution coefficients of its parent object k, which are incorporated into its feature vector $\boldsymbol{h}_i^{(0),\text{features}}$. This modification is formally expressed in Supplementary Equation 2.

$$\boldsymbol{h}_i^{(0),\text{features}} = [\dot{\boldsymbol{x}}_i^t, \|\dot{\boldsymbol{x}}_i^t\|, \dot{\boldsymbol{x}}_i^{t-1}, \|\dot{\boldsymbol{x}}_i^{t-1}\|, \boldsymbol{d}_i^{t_0}, \|\boldsymbol{d}_i^{t_0}\|, \boldsymbol{d}_i^t, \|\boldsymbol{d}_i^t\|, \boldsymbol{b}, m, c_1, c_2] \tag{2}$$
$$\dot{\boldsymbol{x}}_i^t = \boldsymbol{x}_i^t - \boldsymbol{x}_i^{t-1} \qquad \boldsymbol{d}_i^t = \boldsymbol{x}_i^t - \boldsymbol{x}_k^t$$

where $\dot{\boldsymbol{x}}_i^t$ and $\dot{\boldsymbol{x}}_i^{t-1}$ are the finite-difference velocities at time t and $t-1$, $\boldsymbol{x}_i^t$ is the position of node i at time t, $\boldsymbol{x}_k^t$ is the position of object k at time t, $\boldsymbol{d}_i^t$ and $\boldsymbol{d}_i^{t_0}$ are the center-mass distances at time t and t_0, $\boldsymbol{b}$ is the one-hot encoded object type, m is the object's mass, c_1 the friction coefficient and c_2 the restitution coefficient, [] indicates concatenation and $\|\|$ is the Euclidean norm.

The removal of object cells $X^{(4)}$ also affects the structure of the physics-informed message-passing. To ensure a fair comparison and maintain the model's ability to propagate long-range information, we introduce a virtual node at the center of mass of each object. This virtual node is connected to all other nodes belonging to the object with $X^{(1)}$ cells, effectively functioning as a hub for intra-object messages. This approach mimics the virtual node strategy employed in FIGNet [9].



Supplementary Figure 9: Ablated model without object cells $X^{(4)}$. Virtual nodes are introduced at each object's center of mass within $X^{(0)}$ cells to enhance long-range information propagation. An additional node-to-node message-passing step (step 5) is also included, absent from the default model, to further support effective information transfer.

The architecture of the ablated model is depicted in Supplementary Figure 9 and contains 539 651 learnable parameters.

Ablation on physics-informed message-passing To illustrate the importance of incorporating physics inductive biases in our message-passing, we perform ablation experiments where we relax these constraints, allowing learnable messages to propagate without restrictions between all cell types in each layer. Specifically, any cell $x^{(r)}$ can send learnable messages to $x^{(0)}$, $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, and $x^{(4)}$ at any message-passing step, provided they are topologically connected. This contrasts with HOPNet's structured information flow, which enforces physically meaningful pathways. (i.e., first nodes-to-faces, then faces-to-collisions, etc...). The architecture of the ablated model is depicted in Supplementary Figure 10.

However, this unconstrained message-passing does not imply a fully connected combinatorial complex (where every node, edge, or face can communicate arbitrarily). Importantly, the underlying topological structure remains unchanged from the default HOPNet model, with only the message-passing rules altered. HOPNet propagates information sequentially—for example, in step 1 (Figure 3), mesh surfaces embeddings $\boldsymbol{h}^{(2)}$ are updated with messages from nodes $\boldsymbol{m}^{\mathbf{0\to 2}}$, edges $\boldsymbol{m}^{\mathbf{1\to 2}}$, and objects $\boldsymbol{m}^{\mathbf{4\to 2}}$. In contrast, the "free-flowing" variant allows messages to be exchanged between all cell types at any step (e.g., from nodes-to-nodes, nodes-to-edges, ...), removing the structure sequential constraints.

It is important to note that removing these physics-guided pathways and using more layers significantly increases the number of parameters in the model. With hidden embeddings of size 32, three layers of message-passing, and two-layer MLPs, this ablated model with non-sequential message-passing has 951 814 learnable parameters, 34% more than HOPNet. This added complexity does not translate into better performance—as shown in Section 3.4 of the main article—as the absence of physics inductive biases leads to less efficient learning and higher risk of overfitting.

Ablation on physics-based features To illustrate the significance of physics-informed biases in feature selection, we conduct an experiment by removing a critical component of the node features $\boldsymbol{h}^{(0),\text{features}}$: the distance from each node to the object's center of mass. This simplification altering the standard features of a node v_i is formally expressed in Supplementary Equation 3.

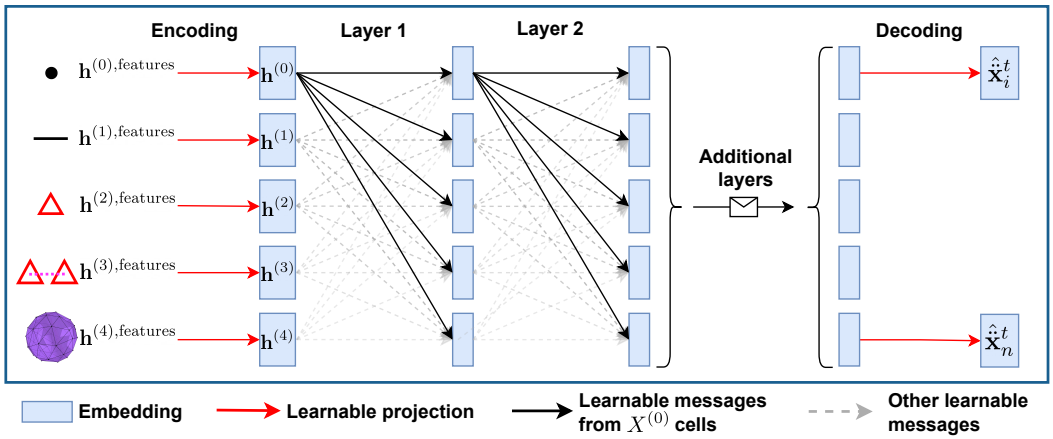
$$\boldsymbol{h}_i^{(0),\text{features}} = [\dot{\boldsymbol{x}}_i^t, \|\dot{\boldsymbol{x}}_i^t\|, \dot{\boldsymbol{x}}_i^{t-1}, \|\dot{\boldsymbol{x}}_i^{t-1}\|] \qquad \dot{\boldsymbol{x}}_i^t = \boldsymbol{x}_i^t - \boldsymbol{x}_i^{t-1} \tag{3}$$

where $\dot{\boldsymbol{x}}_i^t$ and $\dot{\boldsymbol{x}}_i^{t-1}$ are the finite-difference velocities at time t and $t-1$, [] indicates concatenation and $\|\|$ is the Euclidean norm.

The models processing the simplified node features $\boldsymbol{h}^{(0),\text{features}}$, with the center-mass distance removed, contain 706 054 learnable parameters.

3.5 Influence of the model size

To assess how model size influences the ability to learn rigid body dynamics, we developed multiple variants of our standard architecture. One variant increases the expressiveness of the learnable



Supplementary Figure 10: Ablated model without physics-informed message-passing. Learnable messages are enabled between all cell types. To maintain good visibility, only the learnable messages from $X^{(0)}$ cells are shown as solid lines. All other learnable messages are shown as dashed lines with decreasing opacity (purely for visualization purposes).

messages by using 3-layer MLPs instead of the 2-layer MLPs in the standard model. Another variant introduces two layers of physics-informed message-passing, doubling the number of propagation steps compared to the standard approach. Additionally, we explore a more compact model by reducing the hidden embedding size from 128 to 64 across all MLPs. The results from this study are graphically shown in Supplementary Figure 11 and given in tabular format in Supplementary Table 9.

The model with 3-layer MLPs contains $1\,036\,294$ learnable parameters, while the model using two layers of message-passing has $1\,286\,150$ learnable parameters. In contrast, the smaller variant with reduced embedding size has $180\,998$ learnable parameters.

Supplementary Figure 11 illustrates that reducing the model size increases the translation error (11a), which is expected due to the large decrease in learning capacity. Interestingly, the effect on the orientation error (11b) is minimal, with a slight reduction in error. Overall, despite having only 26% of the standard model's parameters, this smaller variant maintains strong accuracy, highlighting the efficiency of our topological representation and physics-informed message-passing architecture.

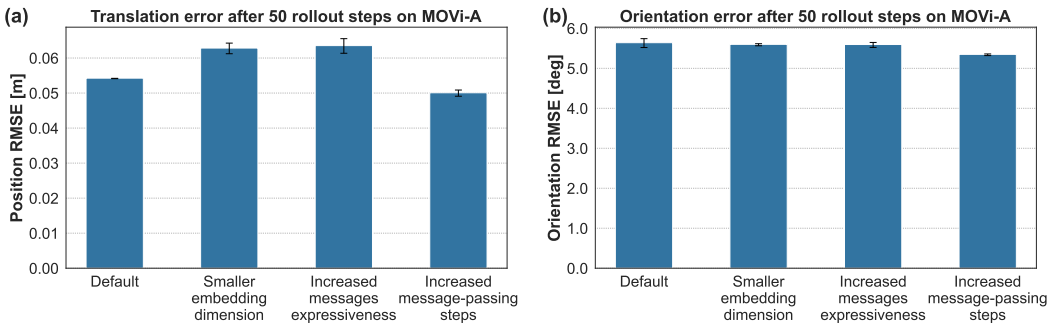
In contrast, increasing message expressiveness by using 3-layer MLPs does not show a clear positive impact. It significantly degrades translation accuracy while improving orientation accuracy, suggesting that the standard 2-layer MLPs are sufficient to capture the non-linear interactions of rigid body dynamics. Finally, introducing additional message-passing steps yields a small boost in overall accuracy, but at the cost of an 81% increase in model size. This trade-off mirrors the design choices seen in baseline models, which employ very deep networks and up to ten message-passing layers, yet exhibit diminishing returns on performance.

Despite these variations in architecture, all of our model variants outperform the baseline models by a significant margin, further demonstrating the robustness and efficiency of our approach.

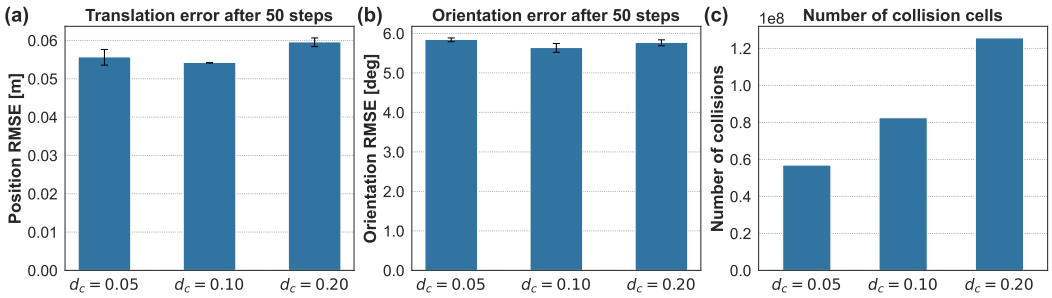
3.6 Influence of the collision radius

To further assess the robustness of our approach to critical hyperparameters, we test our framework's robustness to different collision radius values. The collision radius d_c is the threshold for defining collision contact cells $X^{(3)}$: a collision contact cell $x^{(3)}_{s\leftrightarrow r}$ is created between any two mesh faces $x^{(2)}_s$ and $x^{(2)}_r$ from separate objects if they are within d_c distance of each other.

We trained separate models on the `MOVi-A` dataset using three different collision radii: 0.10 (the default in all experiments); 0.05 (smaller radius, resulting in fewer collision contact cells); and 0.20 (larger radius, resulting in more collision contacts cells). The results are presented in Supplementary Figure 12 and given in tabular format in Supplementary Table 10.



Supplementary Figure 11: Influence of the model size on the autoregressive rollout performance. (**a**) Translation and (**b**) orientation errors after a rollout horizon of $T = 50$ steps on all dynamic objects in the scene. Error bars indicate the mean and standard deviation across three independent random seeds.



Supplementary Figure 12: Influence of the collision radius d_c on the autoregressive rollout performance. (**a**) Translation and (**b**) orientation errors after a rollout horizon of $T = 50$ steps on the `MOVi-A` dataset when using three different collision radii. (**c**) Total number of collision contact cells $X^{(3)}$ in the whole `MOVi-A` dataset for each collision radius. Error bars indicate the mean and standard deviation across three independent random seeds.

As shown in Supplementary Figure 12, our method exhibits strong robustness to changes in d_c: the positional RMSE varies by only 0.005 [m] and the rotational RMSE by 0.205 [deg] between the best and worst results.

Additionally, using a smaller collision radius leads to fewer collision contact cells, with a total of $56\,926\,137$, $82\,505\,837$, and $125\,666\,995$ cells with collision radius 0.05, 0.10, and 0.20 respectively (shown in Supplementary Figure 12c). Less collision contact cells reduce the total computational load.

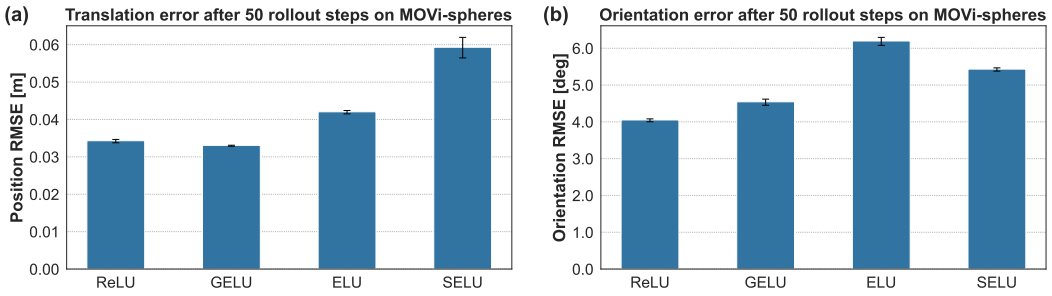
This study, together with our analysis of model size, highlights the exceptional robustness of our framework across critical hyperparameters. The minimal performance variations observed confirm its adaptability and potential for seamless transfer across various tasks, minimizing the need for extensive hyperparameter tuning and making it a versatile tool in diverse applications.

3.7 Influence of the activation function

In addition to the supplementary study on MLP depth in Supplementary Section 3.5, we evaluate the impact of activation function choice within MLPs. The default HOPNet model uses ReLU [11] in all MLPs, as it mitigates vanishing gradients issues more effectively than sigmoid or $\tanh$ [12], while also being computationally efficient.

To assess the impact of alternative activations, we trained HOPNet models on the `MOVi-spheres` dataset using SELU [13], GELU [14], and ELU [15]. These functions have been explored for their ability to improve gradient flow and provide smoother nonlinearities [12], potentially enhancing the expressiveness of our MLPs. All models were trained with the same training hyperparameters (epochs, learning rate, ...) as the default HOPNet model using ReLU.

The results, presented in Supplementary Figure 13 and Supplementary Table 11, indicate that ReLU achieves the best overall performance, followed closely by GELU. While GELU reduces translation



Supplementary Figure 13: Influence of the MLP activation function on the autoregressive rollout performance. (**a**) Translation and (**b**) orientation errors after a rollout horizon of $T = 50$ steps on the `MOVi-spheres` dataset when using four different activation functions. Error bars indicate the mean and standard deviation across three independent random seeds.

error by 3.61%, it increases orientation error by 12.16%. ELU and SELU perform significantly worse, with translation 22.54 and 72.85% higher respectively, and orientation errors 52.96 and 34.09% higher. Additionally, ELU and SELU exhibit greater sensitivity to initialization, as reflected in their larger standard deviation across three random seeds.

One possible reason for ReLU's superior performance is its sharp transition between zero and positive activation, which mirrors the abrupt and highly non-linear nature of collisions: either an object is in contact or it is not. In contrast, smoother activation functions like GELU and SELU introduce gradual transitions, which may not capture the sudden changes in dynamics during collisions as effectively.

3.8 Videos of rollouts

We provide videos of autoregressive rollouts for all standard models (S, A, and B) across the four datasets (`MOVi-spheres`, `MOVi-A`, `MOVi-B`, and `MOVi-C`), available at this link. Each video covers a rollout of 240 timesteps and is played back at 120 FPS (half of the simulation rate) to improve visibility of the dynamics.

3.9 Detailed results as tables

For clarity and ease of comparison, we present the results from the main text as tables. Supplementary Tables 4 and 5 summarize the performance comparisons with baseline models on the `MOVi-A` and `MOVi-B` datasets, respectively. Supplementary Table 6 provides the results from the generalization experiments, while Supplementary Table 8 details the outcomes of the ablation studies on physics-informed biases and topological representations.

Supplementary Table 4: Performance comparison with baseline models on the `MOVi-A` dataset. Detailed results corresponding to Section 3.1 of the main article. The mean and standard deviation are computed over three random seeds.

Model	Metric	25 steps	50 steps	75 steps	100 steps
MGN+	RMSE^{pos}	N/A	0.705 ± 0.069	N/A	N/A
	RMSE^{ori}	N/A	31.210 ± 0.989	N/A	N/A
MGN-LargeRadius+	RMSE^{pos}	N/A	0.119 ± 0.009	N/A	N/A
	RMSE^{ori}	N/A	15.069 ± 0.649	N/A	N/A
FIGNet	RMSE^{pos}	N/A	0.115 ± 0.008	N/A	N/A
	RMSE^{ori}	N/A	14.837 ± 0.175	N/A	N/A
Our framework	RMSE^{pos}	0.014 ± 10^{-4}	0.054 ± 10^{-4}	0.115 ± 0.002	0.196 ± 0.003
	RMSE^{ori}	1.634 ± 0.047	5.641 ± 0.222	11.842 ± 0.371	18.828 ± 0.489

Supplementary Table 5: Performance comparison with baseline models on the `MOVi-B` dataset. Detailed results corresponding to Section 3.1 of the main article. The mean and standard deviation are computed over three random seeds.

Model	Metric	25 steps	50 steps	75 steps	100 steps
MGN*	RMSE^{pos}	N/A	0.538 ± 0.035	N/A	N/A
	RMSE^{ori}	N/A	26.914 ± 0.783	N/A	N/A
MGN-LargeRadius*	RMSE^{pos}	N/A	0.460 ± 0.045	N/A	N/A
	RMSE^{ori}	N/A	26.342 ± 1.397	N/A	N/A
FIGNet	RMSE^{pos}	N/A	0.127 ± 0.006	N/A	N/A
	RMSE^{ori}	N/A	13.990 ± 0.464	N/A	N/A
Our framework	RMSE^{pos}	0.010 ± 10^{-4}	0.047 ± 0.001	0.101 ± 0.003	0.176 ± 0.007
	RMSE^{ori}	1.064 ± 0.013	4.911 ± 0.004	10.350 ± 0.032	17.912 ± 0.123

Supplementary Table 6: Generalization performance across `MOVi-spheres`, `MOVi-A`, and `MOVi-B`. Detailed results corresponding to Section 3.2 of the main article. The mean and standard deviation are computed over three random seeds.

Training	Testing	Metric	25 steps	50 steps	75 steps	100 steps
MOVi-spheres	MOVi-spheres	RMSE^{pos}	0.007 ± 10^{-4}	0.034 ± 10^{-4}	0.073 ± 0.002	0.124 ± 0.005
		RMSE^{ori}	0.913 ± 0.018	4.051 ± 0.078	8.212 ± 0.200	13.675 ± 0.317
	MOVi-A	RMSE^{pos}	0.014 ± 10^{-4}	0.050 ± 0.001	0.112 ± 0.004	0.197 ± 0.007
		RMSE^{ori}	1.534 ± 0.005	5.300 ± 0.043	11.287 ± 0.107	17.740 ± 0.109
	MOVi-B	RMSE^{pos}	0.010 ± 10^{-4}	0.049 ± 0.001	0.106 ± 0.003	0.188 ± 0.006
		RMSE^{ori}	0.973 ± 0.015	4.577 ± 0.051	9.751 ± 0.118	17.134 ± 0.199
MOVi-A	MOVi-spheres	RMSE^{pos}	0.010 ± 10^{-4}	0.046 ± 0.003	0.100 ± 0.008	0.172 ± 0.012
		RMSE^{ori}	1.057 ± 0.020	4.551 ± 0.144	9.159 ± 0.372	15.165 ± 0.595
	MOVi-B	RMSE^{pos}	0.011 ± 10^{-4}	0.055 ± 0.002	0.117 ± 0.006	0.202 ± 0.010
		RMSE^{ori}	1.109 ± 0.038	5.093 ± 0.132	10.769 ± 0.312	18.656 ± 0.571
MOVi-B	MOVi-spheres	RMSE^{pos}	0.010 ± 10^{-4}	0.045 ± 0.003	0.095 ± 0.006	0.160 ± 0.011
		RMSE^{ori}	1.058 ± 0.008	4.583 ± 0.060	9.090 ± 0.162	15.043 ± 0.268
	MOVi-A	RMSE^{pos}	0.014 ± 10^{-4}	0.055 ± 0.003	0.120 ± 0.008	0.202 ± 0.015
		RMSE^{ori}	1.644 ± 0.037	5.716 ± 0.141	12.079 ± 0.266	19.148 ± 0.412

Supplementary Table 7: Generalization performance on the `MOVi-C` dataset. Detailed results corresponding to Section 3.2 of the main article. The mean and standard deviation are computed over three random seeds.

Training	Metric	25 steps	50 steps	75 steps	100 steps
MOVi-spheres	RMSE^{pos}	0.026 ± 10^{-4}	0.104 ± 0.001	0.234 ± 0.002	0.407 ± 0.003
	RMSE^{ori}	1.777 ± 0.009	6.564 ± 0.031	13.569 ± 0.068	21.891 ± 0.133
MOVi-A	RMSE^{pos}	0.027 ± 10^{-4}	0.109 ± 0.001	0.245 ± 0.005	0.425 ± 0.009
	RMSE^{ori}	1.906 ± 0.025	6.958 ± 0.075	14.219 ± 0.100	22.845 ± 0.111
MOVi-B	RMSE^{pos}	0.026 ± 10^{-4}	0.105 ± 10^{-4}	0.238 ± 10^{-4}	0.410 ± 0.002
	RMSE^{ori}	1.926 ± 0.015	7.094 ± 0.034	14.558 ± 0.049	23.431 ± 0.095

Supplementary Table 8: Results from ablation study on topological representation and physics-informed Message Passing (MP). Detailed results corresponding to Section 3.4 of the main article. The mean and standard deviation are computed over three random seeds.

Model	Dataset	Metric	25 steps	50 steps	75 steps	100 steps
No object cells $X^{(4)}$	MOVi-A	RMSE^{pos}	0.019 ± 0.001	0.078 ± 0.006	0.186 ± 0.015	0.337 ± 0.028
		RMSE^{ori}	$1.220 \pm 0..033$	4.746 ± 0.248	10.903 ± 0.570	18.359 ± 0.570
No center-mass distance	MOVi-spheres	RMSE^{pos}	0.009 ± 0.001	0.040 ± 0.003	0.085 ± 0.007	0.145 ± 0.012
		RMSE^{ori}	0.767 ± 10^{-5}	3.582 ± 10^{-4}	7.339 ± 10^{-4}	12.406 ± 10^{-4}
	MOVi-A	RMSE^{pos}	0.017 ± 10^{-4}	0.068 ± 0.004	0.145 ± 0.012	0.245 ± 0.024
		RMSE^{ori}	1.429 ± 0.003	4.971 ± 0.017	10.575 ± 0.023	16.613 ± 0.056
Non-sequential MP	MOVi-A	RMSE^{pos}	0.016 ± 10^{-4}	0.061 ± 0.003	0.133 ± 0.004	0.227 ± 0.008
		RMSE^{ori}	1.287 ± 0.015	4.557 ± 0.076	9.975 ± 0.171	16.672 ± 0.325

Supplementary Table 9: Results from ablation study on model size on the `MOVi-A` dataset. Detailed results corresponding to Supplementary Section 3.5. The mean and standard deviation are computed over three random seeds.

Model	Dataset	Metric	25 steps	50 steps	75 steps	100 steps
3-layers MLPs	MOVi-A	RMSE^{pos}	0.017 ± 0.001	0.063 ± 0.004	0.134 ± 0.008	0.225 ± 0.013
		RMSE^{ori}	1.600 ± 0.042	5.595 ± 0.124	11.653 ± 0.325	18.349 ± 0.344
Two MP layers	MOVi-A	RMSE^{pos}	0.013 ± 0.001	0.050 ± 0.001	0.109 ± 0.002	0.187 ± 0.004
		RMSE^{ori}	1.544 ± 0.014	5.349 ± 0.041	11.300 ± 0.061	17.675 ± 0.070
64-embeddings	MOVi-A	RMSE^{pos}	0.017 ± 10^{-4}	0.062 ± 0.003	0.132 ± 0.007	0.224 ± 0.011
		RMSE^{ori}	1.608 ± 0.011	5.600 ± 0.051	11.831 ± 0.090	18.669 ± 0.206

Supplementary Table 10: Results from ablation study on collision radius d_c on the `MOVi-A` dataset. Detailed results corresponding to Supplementary Section 3.6. The mean and standard deviation are computed over three random seeds.

Collision radius	Dataset	Metric	25 steps	50 steps	75 steps	100 steps
$d_c = 0.05$	MOVi-A	RMSE^{pos}	0.015 ± 10^{-4}	0.055 ± 0.004	0.117 ± 0.006	0.199 ± 0.011
		RMSE^{ori}	1.692 ± 0.037	5.846 ± 0.095	12.324 ± 0.142	19.451 ± 0.206
$d_c = 0.10$	MOVi-A	RMSE^{pos}	0.014 ± 10^{-4}	0.054 ± 10^{-4}	0.115 ± 0.002	0.196 ± 0.004
		RMSE^{ori}	1.634 ± 0.047	5.641 ± 0.222	11.841 ± 0.370	18.826 ± 0.504
$d_c = 0.20$	MOVi-A	RMSE^{pos}	0.015 ± 10^{-4}	0.059 ± 0.002	0.126 ± 0.005	0.213 ± 0.008
		RMSE^{ori}	1.655 ± 0.045	5.771 ± 0.150	12.196 ± 0.229	19.341 ± 0.391

Supplementary Table 11: Results from supplemental study on the activation function used in MLPs on the `MOVi-spheres` dataset. Detailed results corresponding to Supplementary Section 3.7. The mean and standard deviation are computed over three random seeds.

Activation function	Dataset	Metric	25 steps	50 steps	75 steps	100 steps
ReLU [11]	MOVi-spheres	RMSE^{pos}	0.007 ± 10^{-4}	0.034 ± 10^{-4}	0.073 ± 0.002	0.124 ± 0.005
		RMSE^{ori}	0.913 ± 0.018	4.051 ± 0.078	8.212 ± 0.200	13.675 ± 0.317
GELU [14]	MOVi-spheres	RMSE^{pos}	0.007 ± 10^{-4}	0.033 ± 10^{-4}	0.071 ± 0.001	0.122 ± 0.002
		RMSE^{ori}	0.996 ± 0.031	4.544 ± 0.166	9.579 ± 0.430	16.083 ± 0.794
ELU [15]	MOVi-spheres	RMSE^{pos}	0.009 ± 10^{-4}	0.042 ± 10^{-4}	0.091 ± 0.002	0.159 ± 0.007
		RMSE^{ori}	1.343 ± 0.047	6.197 ± 0.217	13.498 ± 0.515	22.780 ± 0.827
SELU [13]	MOVi-spheres	RMSE^{pos}	0.014 ± 0.001	0.059 ± 0.005	0.127 ± 0.012	0.217 ± 0.020
		RMSE^{ori}	1.272 ± 0.013	5.432 ± 0.088	11.069 ± 0.232	18.194 ± 0.478

Supplementary References

[1] Greff, K. *et al.* Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 3749–3761 (2022).

[2] Coumans, E. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*, SIGGRAPH '15 (Association for Computing Machinery, New York, NY, USA, 2015).

[3] Downs, L. *et al.* Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, 2553–2560 (IEEE, 2022).

[4] Community, B. O. *Blender - A 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam (2018).

[5] Sanchez-Gonzalez, A. *et al.* Learning to simulate complex physics with graph networks. In III, H. D. & Singh, A. (eds.) *Proceedings of the 37th International Conference on Machine Learning*, vol. 119 of *Proceedings of Machine Learning Research*, 8459–8468 (PMLR, 2020).

[6] Ba, J. L., Kiros, J. R. & Hinton, G. E. Layer normalization (2016). 1607.06450.

[7] Batzner, S. *et al.* E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications* **13**, 2453 (2022).

[8] Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)* (2015).

[9] Allen, K. R. *et al.* Learning rigid dynamics with face interaction graph networks. In *International Conference on Learning Representations (ICLR)* (2023).

[10] Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A. & Battaglia, P. W. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations (ICLR)* (2021).

[11] Glorot, X., Bordes, A. & Bengio, Y. Deep sparse rectifier neural networks. In Gordon, G., Dunson, D. & Dudík, M. (eds.) *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, vol. 15 of *Proceedings of Machine Learning Research*, 315–323 (PMLR, Fort Lauderdale, FL, USA, 2011).

[12] Apicella, A., Donnarumma, F., Isgrò, F. & Prevete, R. A survey on modern trainable activation functions. *Neural Networks* **138**, 14–32 (2021).

[13] Klambauer, G., Unterthiner, T., Mayr, A. & Hochreiter, S. Self-normalizing neural networks. *Advances in neural information processing systems* **30** (2017).

[14] Hendrycks, D. & Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).

[15] Clevert, D.-A., Unterthiner, T. & Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* (2015).