HEIR: Learning Graph-Based Motion Hierarchies

Cheng Zheng[1*] **William Koch**[1*] **Baiang Li**[1] **Felix Heide**[1,2]
[1]Princeton University [2]Torc Robotics
chengzh, william.koch, baiang.li, fheide@princeton.edu

Abstract

Hierarchical structures of motion exist across research fields, including computer vision, graphics, and robotics, where complex dynamics typically arise from coordinated interactions among simpler motion components. Existing methods to model such dynamics typically rely on manually-defined or heuristic hierarchies with fixed motion primitives, limiting their generalizability across different tasks. In this work, we propose a general hierarchical motion modeling method that learns structured, interpretable motion relationships directly from data. Our method represents observed motions using graph-based hierarchies, explicitly decomposing global absolute motions into parent-inherited patterns and local motion residuals. We formulate hierarchy inference as a differentiable graph learning problem, where vertices represent elemental motions and directed edges capture learned parent-child dependencies through graph neural networks. We evaluate our hierarchical reconstruction approach on three examples: 1D translational motion, 2D rotational motion, and dynamic 3D scene deformation via Gaussian splatting. Experimental results show that our method reconstructs the intrinsic motion hierarchy in 1D and 2D cases, and produces more realistic and interpretable deformations compared to the baseline on dynamic 3D Gaussian splatting scenes. By providing an adaptable, data-driven hierarchical modeling paradigm, our method offers a formulation applicable to a broad range of motion-centric tasks. *Project Page:* https://light.princeton.edu/HEIR/

1 Introduction

Many natural and artificial motion systems comprise simple motion primitives that coordinate to produce complex behaviors. Understanding these structured relationships — commonly called motion hierarchies — is fundamental across multiple research areas, from action recognition in computer vision [5] to verifiable prediction in robotics [1, 31]. By capturing multi-scale dependencies, hierarchical models tame the combinatorial explosion when generating, predicting, or controlling motion.

In *computer vision*, hierarchical representations decompose raw trajectories into semantically meaningful layers. Early work introduced fixed hierarchies for activity recognition and motion capture [5, 1, 31], while more recent methods learn multi-level encodings that bridge low-level displacements and high-level actions [18, 19]. These approaches identify repeated motion patterns and enable reasoning over sub-actions, improving tasks like video generation and pose estimation. In *robotics*, hierarchical structures guide both planning and execution. Global objectives — such as navigating to a goal — are decomposed into coordinated limb or joint movements via modular controllers. Hierarchical reinforcement learning architectures separately optimize high-level navigation and local motor policies [10], and modular schemes coordinate limb-specific controllers under whole-body objectives [42]. This multi-scale decomposition enhances adaptability and robustness in dynamic environments.

*These authors contributed equally to this work. Listing order is random.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Recent advances move beyond manually defined templates toward data-driven hierarchy discovery. Hierarchical VAEs introduce multi-level latent spaces that explicitly model coarse global trajectories and fine local adjustments [20]. Diffusion-based frameworks employ semantic graphs to generate motions at successive abstraction levels, offering fine control without hand-crafted priors [13]. These methods aim to learn interpretable, generalizable motion representations directly from data, closing the gap between fixed kinematic models and the rich variability of real-world behaviors.

Despite differences in objectives and formulations, these research areas still face common technical challenges in modeling motion hierarchies. Most approaches represent motion structures either with hand-crafted heuristics or non-interpretable neural modules [21, 32, 37, 18], making them difficult to transfer across tasks. These models also struggle to find the appropriate coarseness, i.e., balancing the granularity of motion primitives against the expressiveness needed for the task. As such, automatically discovering interpretable, transferable representations of motion hierarchies is an open challenge, requiring adaptively selecting the right level of abstraction for diverse downstream applications.

To address these challenges, we *propose* a general-purpose hierarchical motion modeling method that learns interpretable, structured motion relationships through a graph-based network. We introduce a learnable motion graph, where vertices correspond to subsets of discrete motion elements (e.g., Gaussian splats, tracked keypoints), and directed edges define a learned hierarchy of motion dependencies. To enable this, we learn edge weights on a proximity graph to infer parent-child relationships: the resulting edge weights define the parent probability distribution for any motion element, from which we sample a discrete motion hierarchy. This allows us to model both global constraints and local variations in a unified, data-driven way.

We *validate* the generality and effectiveness of our method on three challenging tasks: two synthetic benchmarks involving structured point motion to show expressiveness (one translational and one rotational), as well as dynamic 3D Gaussian splatting for high-fidelity scene deformation. In all settings, the proposed approach effectively captures and exploits the underlying hierarchical structures. Across representative dynamic scenes, our method achieves improvements over existing methods for all scenarios in scene deformation for pixel error and perceptual metrics, highlighting its effectiveness in producing perceptually and structurally faithful deformations.

2 Related Work

We review relevant work in two areas: motion representations and 3D scene deformation.

Hierarchical Motion Representation and Decomposition. Motion representations capture structure in dynamic systems to support analysis, generation, and control. We existing works have explored explicit or implicit motion models.

Conventional explicit approaches impose predefined structure on motion to improve control, analysis or generation. In humanoid settings, skeletons define natural hierarchical relationships which have been used for various downstream tasks such as SDF learning or motion re-targeting [2]. For video understanding, motion programs [18] represent human behavior as sequences of symbolic motion primitives. The FineGym dataset [27] takes another approach and develops a three-layer semantic hierarchy of pre-defined actions, which has been shown to be useful in action recognition [27, 19]. These methods are largely limited to a domain, however, as defining an explicit structure relies on domain-specific assumptions.

On the implicit side, unsupervised methods such as spectral clustering [5] and Moving Poselets [32] learn task-specific motion patterns that improve recognition without relying on predefined semantics. More closely related to our work, in physics-based settings, relationships emerge by learning how entities interact with each other [6, 26]. Similarly, Neural Relational Inference [16] recovers the underlying interaction graph even when the connectivity is unknown a priori.

At the intersection, we find methods that rely on explicit structures, but learn the relevant components. MovingParts [38] factorizes dynamic NeRF scenes into rigidly moving segments for part-level editing, and an unsupervised video approach clusters pixels by principal motion axes to animate articulated objects [28]. Both recover meaningful parts but also do not infer parent–child dependencies, leaving the full hierarchy unspecified. Unlike prior work, *we do not assume any prior domain*, dimensionality or naturally occurring structures in the data. *Our method is capable of decomposing the motion on its own* - at the same time, it provides an interpretable and controllable structure to the motion.

3D Scene Deformation and Editing. Traditional methods for scene deformation explicitly define geometric transformations using mesh-based deformation energies or cage-based constraints. Approaches such as Laplacian coordinates [29, 30] preserve local geometric details by formulating deformation as energy minimization. Cage-based methods [24, 14, 40] encapsulate an object within a coarse control mesh, allowing intuitive deformation through sparse user manipulation. Although effective for explicit geometric editing, these methods typically assume structured mesh representations.

Recent progress in 3D scene deformation and editing has been largely driven by the emergence of neural implicit representations [22]. NeMF [8] proposes a continuous spatio-temporal representation to model motion as a continuous function over time, enabling smooth interpolation and editing within neural fields. MovingParts [38] discovers object parts from motion cues in dynamic radiance fields for part-level animation. NeRFShop [11] integrates cage-based transformations within NeRFs, facilitating user-driven interactive deformations. Similarly, Wang *et al.* [34] use coarse mesh guidance to impose semantically controllable deformations on neural representations. These approaches primarily focus on achieving visually coherent edits but neglect the underlying hierarchical motion structure.

A few existing methods are addressing structured motion explicitly. CAMM [17] models mesh dynamics using kinematic chains but is limited to occlusion-free settings and mesh-based priors. SC-GS [9] proposes sparse control points for deforming Gaussian splats, yet the neural network-based mapping from control points to Gaussian splats causes entangled deformation effects. Explicit hierarchical modeling of motion in 3D editing remains relatively underexplored, but there are some concurrent efforts. HiMoR [21] employs a manually defined motion tree with a pre-set basis to decompose motion, lacking the flexibility to discover scene-dependent hierarchy. MB-GS [43] represents Gaussian splat dynamics using sparse motion graphs with dual quaternion skinning and learnable weight painting. While providing more structured control, its motion structure is still predefined per-object rather than learned from data.

In contrast to these works, our method learns a hierarchical motion representation from observed scene dynamics. By inferring structured, data-driven parent-child relationships between motion elements, our method achieves interpretable, flexible, and consistent scene deformation and editing.

3 Hierarchical Motion Learning

In this section, we introduce the proposed hierarchical motion learning approach. We first define the problem setup in Section 3.1, then briefly overview our method in Section 3.2. Next, we describe the details in learning of motion hierarchies based on a proximity graph in Section 3.3, and show it can also be extended to rotational motion in Section 3.4. Following, in Section 3.5, we describe how we apply our method to the deformation 3D Gaussian Splat scenes.

3.1 Problem Setup

We tackle a new problem of hierarchical motion modeling that decomposes observed absolute motion into parent-inherited motion and residual local motion, structured by a learnable directed graph hierarchy. Figure 1 provides a graphical reference to this problem. Given N-dim motion elements with observed positions $\mathbf{X}^t \in \mathbb{R}^{N\times d}$ on d-dim over time steps $t = 0, \ldots, T$, we define the absolute velocity or deformation at time step t as the frame-to-frame difference between consecutive time steps $\mathbf{\Delta}^t = \mathbf{X}^{t+1} - \mathbf{X}^t \in \mathbb{R}^{N\times d}$. Our objective is to infer a hierarchy matrix $H \in \{0, 1\}^{N\times N}$, such that absolute deformations $\mathbf{\Delta}^t$ can be decomposed into a parent-inherited motion and a relative motion $\boldsymbol{\delta}^t$ as

$$\mathbf{\Delta}^t = H\mathbf{\Delta}^t + \boldsymbol{\delta}^t. \tag{1}$$

Specifically, $H_{ij} = 1$ means element j is the parent of element i. To make the above equation valid, the hierarchy matrix H should satisfy the following: (1) Each element has exactly one parent, meaning each row has only one non-zero entry; (2) The element cannot be the parent of itself, so the diagonal entries are zero; (3) The hierarchy must not contain cycles, meaning there exists no sequence of distinct indices $i_1, i_2, \ldots, i_k$ (with $k > 1$), such that $H_{i_1 i_2} = H_{i_2 i_3} = \cdots = H_{i_k i_1} = 1$.

The hierarchy matrix H here exactly represents a directed acyclic graph (DAG), which consists of vertices and edges with each edge directed from one vertex to another, such that following those directions will never form a closed loop. Note that the hierarchy matrix defined here is an adjacency matrix with binary values [23, 41], where 1 indicates an edge between the two vertices and 0 otherwise.

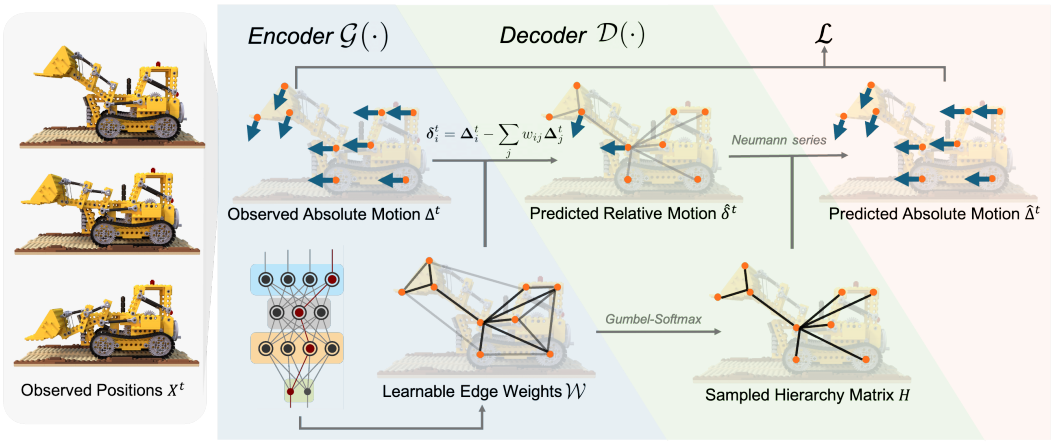

Figure 1: **Learning Motion Hierarchies.** Given a sequence of observed positions $\mathbf{X}^t$ over time (left), we predict absolute motions and candidate graphs based on local spatial proximity. A graph neural network processes this structure to predict edge weights to infer a probabilistic parent-child hierarchy over motion elements (bottom path). The encoder computes the prediction of the relative motion based on these weighted parent candidates. The absolute motion of each motion element is then recursively aggregated from its parent using a residual composition process (top path) and a hierarchy matrix sampled from the edge weights using Gumbel-Softmax. We learn the hierarchy by minimizing the difference between the observed and predicted absolute motions across all time steps.

Other adjacency matrices might have values between 0 and 1 to represent the weight (or cost) of the edge between two vertices.

3.2 Overview

We introduce a hierarchy motion learning method to tackle the above problem, illustrated in Fig. 1.

We define a proximity directed graph $G_0 = (\mathcal{V}, \mathcal{E}_0)$ to learn the optimal hierarchy matrix H. The vertices of the graph $\mathcal{V} = \{\mathbf{\Delta}_i^t\}_{i=1}^N \in \mathbb{R}^{N\times d}$ denotes the set of N motion elements. For each vertex i, we define its neighborhood vertices $\mathcal{N}_k(i) \subset \mathcal{V}$ as the set of its k nearest neighbors in Euclidean space. This neighborhood serves as the parent candidate of the vertex, where $(i, j) \in \mathcal{E}_0$ if and only if $\mathbf{\Delta}_j^t \in \mathcal{N}_k(i)$. The attributes of the edge w_{ij} represent the possibility that vertex j could serve as the parent of vertex i. We obtain the predicted value of relative motions $\hat{\boldsymbol{\delta}}^t$ via message passing and aggregation in the graph, and reconstruct the full motion $\hat{\mathbf{\Delta}}^t$ using H sampled from a normalized distribution of $\mathcal{W}$. The corresponding optimization problem is then to find the edge weights $\mathcal{W}^\star$ that minimize the difference between the predicted motion and ground truth motion

$$\mathcal{W}^\star = \arg\min_{\mathcal{W}} \sum_{t=0}^{T-1} \mathcal{L}_{\text{base}} \left(\mathbf{\Delta}^t, \mathcal{D}\left(\mathcal{G}\left(\mathbf{\Delta}^t; \mathcal{W}\right), H\right)\right), \tag{2}$$

where $\mathcal{G}(\cdot)$ denotes the message passing, aggregation, and vertex update in the graph, and $\mathcal{D}(\cdot)$ represents the decoding from relative motion to absolute motion.

3.3 Learning Motion Hierarchies

We next describe our model structure and the training objective. The model contains an encoder module $\mathcal{G}(\cdot)$ and a decoder module $\mathcal{D}(\cdot)$ that we propose to learn the motion hierarchies. We use a graph neural network as the encoder that takes the observed absolute motions to predict local dynamics $\hat{\boldsymbol{\delta}}^t = \mathcal{G}(\mathbf{\Delta}^t; \mathcal{W})$, and define a decoder module to reconstruct the global dynamics $\hat{\mathbf{\Delta}}^t = \mathcal{D}\left(\hat{\boldsymbol{\delta}}^t, H\right)$ given a hierarchical relationship H and local dynamics. The hierarchy matrix H is sampled from $\mathcal{W}$ in the encoder and passed into the decoder for absolute motion extraction. We supervise this model to learn motions faithfully and explain the observed deformations $\mathbf{\Delta}^t$ across all $t = 0, \ldots, T-1$.

Sparse Message Passing (Encoder $\mathcal{G}$). With the absolute-velocity field $\mathbf{\Delta}^t = [\mathbf{\Delta}_1^t, \ldots, \mathbf{\Delta}_N^t]^\top$ as vertex features, a single message-passing layer operates on the proximity graph G_0. We compute a learnable logit for each edge $(ij) \in \mathcal{E}_0$ based on the input features $\mathbf{\Delta}_i$ and $\mathbf{\Delta}_j$. Specifically, we use a graph attention layer [33] that applies a LeakyReLU to the dot product of a learnable attention vector with the concatenated linear projections of input vertex features, followed by a softmax normalization over all the j's that are in the neighborhood of vertex i as

$$w_{ij} = \text{softmax}_{j \in \mathcal{N}(i)}\big(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{\Delta}_i \| \mathbf{W}\mathbf{\Delta}_j])\big), \tag{3}$$

where $\mathbf{a}$ is the attention vector and $\mathbf{W}$ is the weighted matrix for linear transform. The relative velocity of the vertex in G_0 can then be estimated as the difference between the absolute velocity of themselves, and the weighted subtraction of the absolute velocities of their parent candidates, i.e., $\boldsymbol{\delta}_i^t = \mathbf{\Delta}_i^t - \sum_j w_{ij}\mathbf{\Delta}_j^t$.

Sampling Hierarchies. During training, we draw S candidate hierarchy matrices $\{H^{(s)} \in \{0,1\}^{N\times N}\}_{s=1}^S$ from learned edge weights $\mathcal{W}$. To ensure differentiability in the discrete space, we apply the Gumbel-Softmax trick [12]. For each vertex i, we sample noise $g_{ik}^{(s)} \sim \text{Gumbel}(0,1)$ and compute soft parent probabilities as

$$\tilde{w}_{ik}^{(s)} = \exp\big((w_{ik} + g_{ik}^{(s)})/\tau\big) \Big/ \sum_{j \in \mathcal{N}(i)} \exp\big((w_{ij} + g_{ij}^{(s)})/\tau\big), \tag{4}$$

where $\tau > 0$ is a temperature annealed during training. For the forward pass, we use the hard (straight-through) variant, setting $H_{ij}^{(s)} = 1$ if $j = \arg\max_k \tilde{w}_{ik}^{(s)}$, and 0 otherwise. At the same time, gradients are back-propagated through the soft weights $\tilde{w}^{(s)}$. At inference, we use the maximum-likelihood hierarchy, with $H_{ij} = 1$ if and only if $j = \arg\max_k s_{ik}$.

Absolute Motion Reconstruction (Decoder $\mathcal{D}$). Given a hierarchy matrix $H \in \{0,1\}^{N\times N}$ and the predicted relative velocities $\boldsymbol{\delta}^t \in \mathbb{R}^{N\times d}$ at time t, we reconstruct the absolute velocities $\hat{\mathbf{\Delta}}^t$ by accumulating the relative velocities along the hierarchy. Specifically, the absolute velocity can be expressed as the truncated Neumann series $\hat{\mathbf{\Delta}}^t = \sum_{l=0}^{\infty} H^l \boldsymbol{\delta}^t = \sum_{l=0}^{L_{\max}} H^l \boldsymbol{\delta}^t$, where H^l corresponds to ancestor relationships at depth l of the hierarchy, and $L_{\max} \leq N$ is the maximum depth of the tree. In practice, we also include an early stopping condition $\|H^l \boldsymbol{\delta}^t\| < \varepsilon$ for some $\varepsilon \in \mathbb{R}$ in cases where the depth of the tree is lower than $L_{\max}$.

We note that a hierarchy matrix H satisfying our conditions is acyclic and therefore nilpotent, meaning we could write the Neumann series in a closed-form expression $\hat{\mathbf{\Delta}}^t = (I - H)^{-1}$. However, in practice, we rely on the truncated series for stability purposes. This series converges at the deepest branch while remaining well-defined even if the sampling method occasionally produces cycles.

Training Objective. We learn $\mathcal{W}$ by minimizing two ℓ_1 objectives

$$\mathcal{W}^\star = \arg\min_{\mathcal{W}} \left(\sum_{t=0}^{T-1} \left\| \mathcal{D}\left(\mathcal{G}\left(\mathbf{\Delta}^t; \mathcal{W}\right), H\right) - \mathbf{\Delta}^t \right\|_1 + \lambda \sum_{t=0}^{T-1} \left\| \mathcal{G}\left(\mathbf{\Delta}^t; \mathcal{W}\right) \right\|_1 \right), \tag{5}$$

where the first term encourages the reconstructed absolute velocities to match the ground-truth deformations, and the second term regularizes the magnitude of the relative velocity field, incentivizing the model to minimize local velocities and explain motion primarily through parents. Without this regularizer, the model would trivially minimize the reconstruction loss with the solution $\boldsymbol{\delta}^t = \mathbf{\Delta}^t$ and a hierarchy H corresponding to a star topology, bypassing any meaningful hierarchical structure. Here, λ is a hyperparameter to balance these objectives.

3.4 Enabling Rotation Inheritance

With minor modifications, our method is also capable of inheriting rotations. The key idea is to modify the encoder $\mathcal{G}$ to predict the relative velocity in polar coordinates rather than Cartesian coordinates. For each candidate parent–child pair (i, j), the encoder decomposes motion into a radial velocity component $\dot{r}_{ij}^t$ measuring the rate of change in distance $|\mathbf{r}_{ij}^t|$ between nodes, and an angular velocity component $\dot{\theta}_{ij}^t$ capturing the rate of change in orientation of $\mathbf{r}_{ij}^t$. The relative velocity in

Cartesian coordinates $\boldsymbol{\delta}_{ij}^t$ can be easily reconstructed from the polar components. We denote the aggregated values returned by $\mathcal{G}$ as $\widehat{\dot{r}}_i^t$, $\widehat{\dot{\theta}}_i^t$ and $\widehat{\boldsymbol{\delta}}_i^t$. For details, we refer to the supplemental material. These aggregated predictions are obtained as before, by weighting edge contributions with learned attention scores. Given $\mathcal{L}_{\text{base}}$ from equations 2 and 5, we modify our learning objective as follows:

$$\mathcal{W}^\star = \arg\min_{\mathcal{W}} \sum_{t=0}^{T-1} \Bigg(\mathcal{L}_{\text{base}} \left(\boldsymbol{\Delta}^t, \mathcal{D} \left(\mathcal{G} \left(\boldsymbol{\Delta}^t; \mathcal{W} \right), H \right) \right) + \boldsymbol{\lambda}_\Lambda \mathcal{L}_\Lambda(H) + \sum_{i=1}^{N} (\boldsymbol{\lambda}_r \left\| \widehat{\dot{r}}_i^t \right\|_1 + \boldsymbol{\lambda}_\theta \left\| \widehat{\dot{\theta}}_i^t \right\|_1) \Bigg), \tag{6}$$

where λ_r, λ_θ and λ_Λ are hyperparameters for regularizing radial velocity, angular velocity, and connectivity, respectively. The term $\mathcal{L}_\Lambda(H)$ is a connectivity prior based on the graph Laplacian of the symmetrized hierarchy: let $A = \max(H + H^\top, 1)$ and $L = D - A$ with $D = \operatorname{diag}(A\mathbf{1})$. Its second-smallest eigenvalue $\lambda_2(L)$, known as the algebraic connectivity [4], is strictly positive if and only if the graph is connected. We therefore penalize low connectivity with a hinge loss $\mathcal{L}_\Lambda(H) = \text{relu}(\tau_c - \lambda_2(L))$, encouraging well-formed, non-fragmented hierarchies. The graph Laplacian is widely used for analyzing graph connectivity properties.

3.5 3D Scene Deformations and Editing

Next, we describe how the proposed method applies to 3D Gaussian Splat scene deformations. This setting presents a significant challenge due to the large number of Gaussians involved. —Realistic scenes often contain up to hundred thousands of Gaussians, each with spatial and temporal attributes, making scalable and structured deformation non-trivial. We represent motions in a scene using dynamic 3D Gaussian splatting and treat each Gaussian in the scene as a vertex in the graph. At inference time, we deform the 3D scene with learned hierarchy under the as-rigid-as-possible constraints, and use the deformed Gaussians to generate the new scene.

Preliminaries. Gaussian splatting represents a 3D scene using a set of 3D Gaussians, where each Gaussian G_j is defined by parameters $\{\mu_j, \Sigma_j, \sigma_j, c_j\}$ [15]. μ_j is the 3D center location, Σ_j is the 3D covariance matrix, σ_j is opacity, and c_j denotes spherical harmonic coefficients encoding view-dependent colors. The covariance matrix Σ_j can be decomposed as $\Sigma_j = R_j S_j S_j^T R_j^T$, where $R_j \in \mathbb{SO}(3)$ is a rotation matrix parameterized by a quaternion q_j, and $S_j = \operatorname{diag}(s_j) \in \mathbb{R}^{3\times3}$ is a diagonal matrix encoding the axis-aligned scaling.

To model temporal dynamics, we use a deformation field to predict per-Gaussian offsets as a function of time t. These include translations $\delta\mu_j(t)$, quaternion-based rotations $\delta q_j(t)$, and anisotropic scalings $\delta s_j(t)$. The deformation field can be implemented as an implicit neural network, as in 4D-Gaussians [36] and Deformable-GS [39], or as a composition of shared motion bases and per-Gaussian coefficients, as in Shape-Of-Motion [35] and SC-GS [9]. These models are typically trained via photometric reconstruction losses between rendered outputs and ground-truth video frames.

Our approach builds on these advances but introduces an explicit, learnable motion hierarchy among Gaussians. That means, rather than modeling each motion independently or relying on fixed bases, we infer structured parent-child dependencies that enable interpretable and flexible scene deformation.

Scene Deformation. We infer the deformation of the 3D scene based on user-specified inputs and the learned hierarchy matrix H in Sec. 3.2. We select a Gaussian splat G_h, and trace the hierarchy matrix H to get all its descendants $Desc\,(G_h)$. These positions are treated as "handles" with constrained positions from user input (either translation or rotation around the center). The deformation for the rest of the Gaussian splats is calculated via an as-rigid-as-possible (ARAP) solver [30] to preserve local structure. The ARAP solver optimizes the deformed position as well as the rotations of the Gaussian splats by minimizing the energy given defined handles

$$E(G') = \sum_{i=1}^{N} \omega_i \sum_{j \in \mathcal{N}_i} \omega_{ij} \left\| \left(\mathbf{p}_i' - \mathbf{p}_j' \right) - \mathbf{R}_i' \left(\mathbf{p}_i - \mathbf{p}_j \right) \right\|, \tag{7}$$

where the $\mathbf{p}_i$ and $\mathbf{p}_i'$ are the Gaussian center locations before and after optimization, Here, $\mathbf{R}_i'$ is the optimized rigid rotation matrix $\in \mathbb{SO}(3)$, and ω_i and ω_{ij} are the Gaussian- and edge-dependent weights, which are set to 1 and the cotangent weights according to the original paper. We then apply

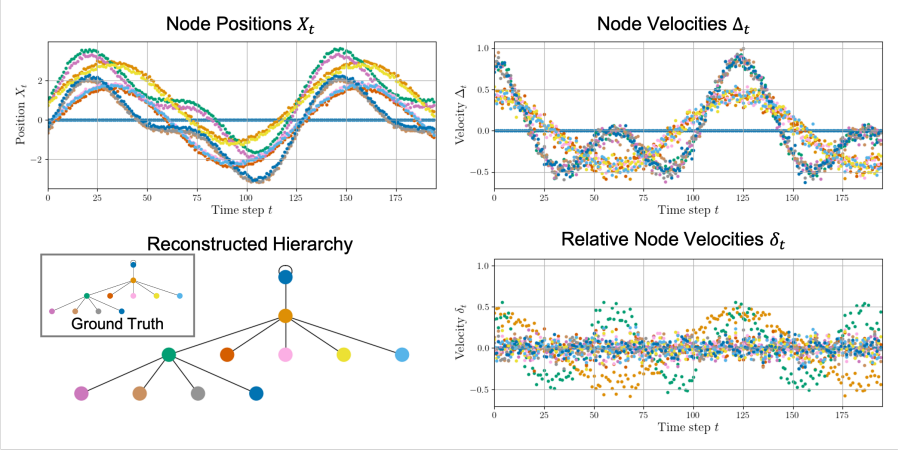

Figure 2: **Learning of Hierarchical Relations in a 1D Trajectory.** We evaluate the proposed hierarchical learning method for a 1D motion trajectory where individual nodes are moving in a hierarchical manner (see Ground Truth motion hierarchy in bottom left inset), but each adding its own unknown motion. Top left to bottom right: (1) raw node positions X_t of the hierarchical trajectories over time, (2) absolute node velocities Δ_t, (3) reconstructed hierarchy from inferred relationships with ground-truth hierarchy in the inset, and (4) relative velocities δ_t with respect to each node parent, given the reconstructed hierarchy (3). We find that the method is able to correctly identify all motions (bottom left) with the two core motions through the orange and green nodes.

the obtained translation $\mathbf{p}'_i - \mathbf{p}_i$ and the rotation matrix $\mathbf{R}'_i$ to the 3D center location and quaternion of the Gaussian, respectively. By re-rendering the scene with updated Gaussian parameters, we obtain an image of the deformed scene. Note that we learn the hierarchy matrix on downsampled Gaussians and use skinning weights to apply deformation to all Gaussians. We illustrate the details of this process in the supplemental material.

4 Experiments

We first validate the proposed method on a toy benchmark that contains 1D motions constructed with known hierarchies, as well as a synthetic planetary orbit dataset. Following, we move to evaluation the method for a high-dimensional task of 3D Gaussian Splat deformations with complex motion patterns.

4.1 1D Toy Example

To validate the expressiveness of the method, we construct a synthetic dataset with known hierarchies. We create a minimal point set $\mathbf{X}^t \subset \mathbb{R}^N$ with $N = 11$ nodes and $t \in \{0, \dots, T\}$, $T = 200$ frames. Node 0 is fixed at the origin ($x_0^t = 0, \forall t$) and serves as the root. Node 1 follows a low-frequency sine motion

$$x_1^t = A_0 \sin(\omega_0 t + \phi_0) + \eta_1^t, \qquad A_0 = 2,\ \omega_0 = 10,\ \eta_1 \sim \mathcal{N}(0, \sigma^2), \tag{8}$$

where $\sigma = 5 \times 10^{-3}$ and $\eta_i^t \sim \mathcal{N}(0, \sigma^2)$ is a perturbation. Node 2 is a child of node 1, and adds a higher-frequency component,

$$x_2^t = x_1^t + A_1 \sin(\omega_1 t + \phi_1) + \eta_2^t, \qquad A_1 = 1,\ \omega_1 = 20. \tag{9}$$

The remaining nodes x_i for $i = 3, \dots, 10$ inherit the low-frequency motion from node 1, and—with a probability of $p = 0.5$—also inherit the high-frequency term from node 2. Their initial positions are drawn uniformly from $[-1, 1]$ and perturbed at each time step by $\eta_i^t \sim \mathcal{N}(0, \sigma^2)$. Fig. 2 top-left shows the trajectories of the nodes over time. By construction, we obtain a ground-truth hierarchy matrix $H^\star$ shown in the bottom-left inset of Fig. 2 with three layers: the root, a low-frequency layer, and a high-frequency layer.

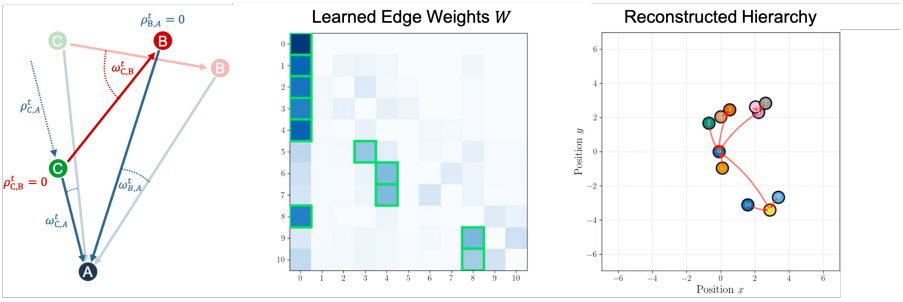

Figure 3: **Learning of hierarchical relations in a planetary system.** We evaluate on a synthetic dataset with rotational hierarchies, a simplified synthetic planetary dataset. From left to right: (1) illustration of the pairwise metrics used for regularization between two timesteps; for clarity, only a subset of possible parent-child relations is shown. Solid arrows indicate potential parent-child vectors, with the color corresponding to the parent candidate. (2) Learned edge weights, where entries with a green border correspond to correct reconstructions. (3) The observed data shown with the reconstructed hierarchy; we note that the "moons" correctly inherit motion from their "planets".

Evaluation. We train 1,000 independent models for 1,000 epochs each, annealing the Gumbel-Softmax temperature τ from 1.5 to 0.3. A custom softmax implementation keeps every parent probability strictly positive, avoiding premature collapse.

After training of the model, we validate the correctness. However, as there are many valid hierarchy reconstructions, we cannot simply match against the ground truth. We validate a hierarchy matrix H by comparing depth-1 parent–child clusters. As they describe the same motion, the same cluster should be found in the ground-truth tree $H^\star$, up to some permutation within the cluster. If this is satisfied for all subgroups, we consider the hierarchy to be valid.

Across all runs we obtain a 73% success rate, indicating that the method reliably recovers the three-layer structure despite noise. We highlight that this is a significant result, as there are 10^{10} potential candidate hierarchies, but only 50 valid ones. There are 5 permutations for the first motion group, 5 for the second as well as 2 potential orderings of the groups.

Method	Accuracy (%)
Proposed	73.0
Monte-Carlo Estimate	5.0×10^{-7}

Table 1: **Hierarchy–Reconstruction Accuracy on the 1D Hierarchical Motion Task.** Models are trained with Gumbel-Softmax annealing and a custom softmax to ensure stable parent assignment. A hierarchy is counted as correct if all depth-1 motion groups match ground-truth clusters up to permutation. Our method recovers a valid hierarchy matrix in 73% of cases, whereas a random Monte-Carlo estimate reconstructs a valid one $\ll 1\%$.

This synthetic dataset demonstrates that our method is capable of disentangling nested motions and discover the correct parent structure, validating the theory before moving to 3D scene deformations. Please refer to the supplement material for additional details.

4.2 Planetary System

To further test the rotational extension, we construct a synthetic planetary system dataset with $N = 11$ nodes and $T = 100$ timesteps. Node 0 represents the star (root), with planets and moons attached as its descendants as shown in Fig. 3. In this synthetic dataset, we assume circular motion and do not consider gravitational influences - the purpose is to show the expressiveness of the system in capturing rotations.

Evaluation. We correctly reconstruct 100% of hierarchies when there is no noise present in the data, and 73.6% with Gaussian noise ($\sigma = 0.05$). In both cases, we train 1,000 independent models for 500 epochs each. Validation against the ground truth hierarchy is easier than in the 1D toy example,

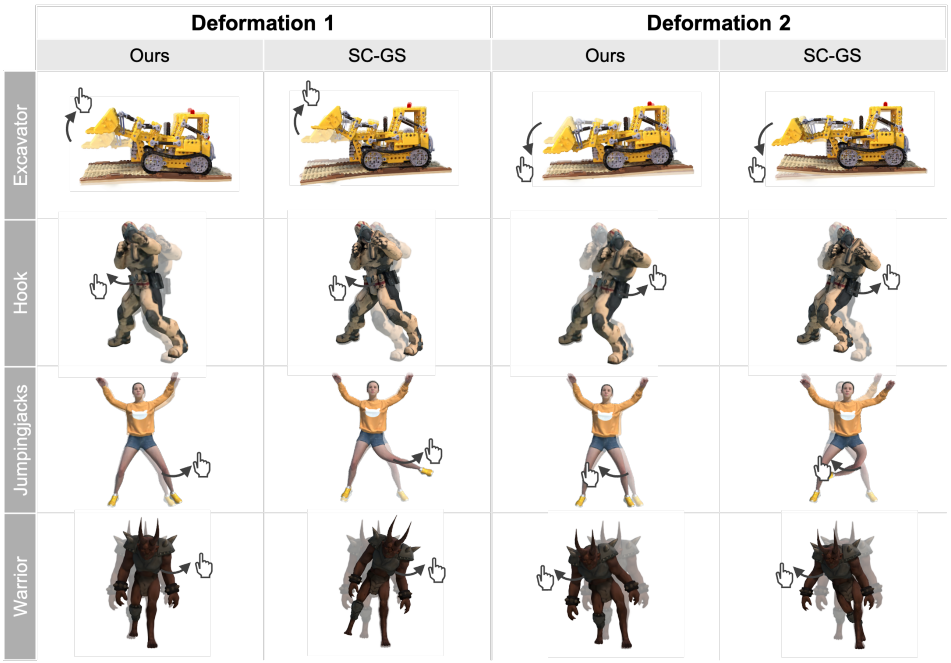

Figure 4: **Qualitative Evaluation of Gaussian Scene Deformation on the D-NeRF [25] dataset.** We evaluate the method for hierarchical relationship learning on Gaussian splitting scenes, with thousands of nodes. Specifically, we show scene deformation for the "Excavator", "Hook", "Jumpingjacks", and "Warrior" scenes from the D-NeRF [9] dataset. The arrows show the user-defined deformation on the faded original scene in two different scenarios. We overlay the resulting deformed scenes for the proposed method and SC-GS [9] on the original scene . The proposed method produces more realistic and physically coherent deformations, preserving structural rigidity, while SC-GS introduces unnatural distortions and misaligned body geometry.

as there is only a single valid hierarchy reconstruction. We use $\lambda_{\dot{r}} = 12.0$ to penalize deviations in distance, $\lambda_{\delta} = 0.8$ to regularize relative velocity, and $\lambda_{\dot{\theta}} = 0.0$ - we do not want to penalize relative angular velocity here, on the contrary. The Laplacian connectivity prior is enforced with weight $\lambda_{\Lambda} = 6.0$. We refer to the supplementary material for additional ablations, in particular on the impact of noise.

4.3 Dynamic Gaussian Splatting Scene Deformation

We next validate the method on 3D dynamic Gaussian splatting with experiments on D-NeRF dataset [25], which contains a variety of rigid and non-rigid deformations of various objects. The scenes are usually represented with hundred thousands of Gaussian with spatial and temporal parameters. We compare to SC-GS [9] as the only very recent method capable of tacking this problem. Note that while there are other recent approaches [21, 43, 7, 3] that tackle this problem, code was not available for any of them at the time of this study. Fig. 4 reports qualitative comparisons to SC-GS [9] with two different interactive deformations on four different D-NeRF scenes.

We find that our method achieves more realistic and coherent deformations, effectively preserving meaningful structural relationships and maintaining scene integrity. Specifically, in the *Excavator* example, SC-GS introduces unnatural bending and distortion on the shovel-body connections and the ground, while our method realistically adjusts the shovel's position only to preserve the excavator's rigid geometry. For the *Hook* example and *Warrior* example, SC-GS produces exaggerated body distortions, whereas our method maintains a plausible body posture and natural limb alignment. Similarly, in the *Jumpingjacks* example, SC-GS generates physically unrealistic leg and arm deformations, whereas our method produces smooth, physically plausible limb movements consistent with human body and motion constraints.

Table 2: **Quantitative Evaluation on the D-NeRF [25] dataset.** We quantitatively assess our method for the task of scene deformation on dynamic scenes from D-NeRF dataset [25]. We evaluate scene reconstruction metrics of known dynamic poses, like shovel lifting, person punching, and jumping. The proposed method improves on existing methods across all scenes on perceptual scene quality and similarity metrics.

	Excavator		Hook		Jumpingjacks		Warrior	
	Ours	SC-GS [9]	Ours	SC-GS [9]	Ours	SC-GS [9]	Ours	SC-GS [9]
PSNR ↑	**21.56**	19.91	**18.3**	15.7	**21.54**	21.12	**16.17**	15.29
SSIM ↑	**0.917**	0.88	**0.93**	0.92	**0.952**	0.944	**0.934**	0.926
CLIP-I ↑	**0.978**	**0.978**	**0.971**	0.958	**0.975**	0.948	**0.985**	0.965
LPIPS ↓	**0.0383**	0.065	**0.0617**	0.0954	**0.0507**	0.0748	**0.0567**	0.089

Table 2 quantifies the quality of the deformed scene using peak signal-to-noise ratio (PSNR), CLIP image-image similarity (CLIP-I), structural similarity (SSIM), and learned perceptual image patch similarity (LPIPS). Note these are not the reconstruction scores against ground-truth views as in the standard D-NeRF benchmark, we compute these metrics by projecting the deformed 3D scenes into 2D images and comparing them against the original scene under the same camera view. Our method consistently outperforms the baseline across all metrics and scenes, indicating improved perceptual and structural fidelity. See the supplement video for the full deformation process of the scenes, and the supplemental material for additional experimental results and ablation experiments.

5 Conclusion

We introduce a general-purpose method for hierarchical motion modeling that learns structured motion relationships directly from data. By formulating motion decomposition as a differentiable graph learning problem, our method infers interpretable parent-child dependencies and disentangles global and local motion behaviors. We validate the method on 1D hierarchical motion reconstruction, a simplified planetary orbit and dynamic 3D scene deformation of Gaussian splitting scenes. We confirm that our approach compares favorably to existing methods and produces more coherent, realistic, and semantically structured deformations. Unlike prior work that relies on fixed motion bases or heuristic hierarchies, our method adapts flexibly to scene dynamics while maintaining interpretability.

Limitations: While our method effectively captures hierarchical motion structure from data, it inherits several limitations from the same data. It depends on the presence and observability of motion in the input and cannot infer latent or task-driven semantics that are not reflected in the motion trajectories—for example, it cannot recover the motion of an object part (e.g., a truck's shovel) if it remains static in the training data. Moreover, the current formulation assumes each motion element has a single parent, which may restrict expressiveness in systems with overlapping or multi-source motion influences.

Despite these limitations, our results suggest that data-driven motion hierarchies offer a promising foundation for structured and generalizable motion modeling. We can strengthen long-range dependency detection by replacing k-NN with global-connectivity variants like sparsely sampled global attention layer, dilated-radius neighbours, or a small set of randomly initialized long-range edges. The learned explicit hierarchy also allows us to selectively add local rigidity during deformation to further avoid unwanted deformation artifacts. We hope this work encourages further exploration of learnable hierarchical representations across domains.

Acknowledgments and Disclosure of Funding

We thank Guangyuan Zhao and Congli Wang for their help with the paper writing, and Jan Philipp Schneider for his input and ideas. Felix Heide was supported by an NSF CAREER Award (2047359), a Packard Foundation Fellowship, a Sloan Research Fellowship, a Disney Research Award, a Sony Young Faculty Award, a Project X Innovation Award, and an Amazon Science Research Award. Cheng Zheng and Felix Heide were supported by a Bosch Research Award.

References

[1] Andrew Benton, Eugen Solowjow, and Prithvi Akella. Verifiable learned behaviors via motion primitive composition: Applications to scooping of granular media. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2549–2555. IEEE, 2024. 1

[2] Sourav Biswas, Kangxue Yin, Maria Shugrina, Sanja Fidler, and Sameh Khamis. Hierarchical neural implicit pose network for animation and motion retargeting. *arXiv preprint arXiv:2112.00958*, 2021. 2

[3] Jiahua Dong and Yu-Xiong Wang. 3DGS-Drag: Dragging gaussians for intuitive point-based 3D editing. In *Proceedings of the International Conference on Learning Representations*, 2025. 9

[4] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973. Publisher: Institute of Mathematics, Academy of Sciences of the Czech Republic. 6

[5] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Activity representation with motion hierarchies. *International Journal of Computer Vision*, 05 2013. 1, 2

[6] Artur Grigorev, Bernhard Thomaszewski, Michael J Black, and Otmar Hilliges. HOOD: Hierarchical graphs for generalized modelling of clothing dynamics. 2023. 2

[7] Xiao Han, Runze Tian, Yifei Tong, Fenggen Yu, Dingyao Liu, and Yan Zhang. ARAP-GS: Drag-driven as-rigid-as-possible 3D gaussian splatting editing with diffusion prior. *arXiv preprint arXiv:2504.12788*, 2025. 9

[8] Chengan He, Jun Saito, James Zachary, Holly Rushmeier, and Yi Zhou. NeMF: Neural motion fields for kinematic animation. *Advances in Neural Information Processing Systems*, 35:4244–4256, 2022. 3

[9] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. SC-GS: Sparse-controlled gaussian splatting for editable dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4220–4230, 2024. 3, 6, 9, 10

[10] Deepali Jain, Atil Iscen, and Ken Caluwaerts. Hierarchical reinforcement learning for quadruped locomotion. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 7551–7557. IEEE, 2019. 1

[11] Clément Jambon, Bernhard Kerbl, Georgios Kopanas, Stavros Diolatzis, Thomas Leimkühler, and George Drettakis. Nerfshop: Interactive editing of neural radiance fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(1), 2023. 3

[12] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. 5

[13] Peng Jin, Yang Wu, Yanbo Fan, Zhongqian Sun, Wei Yang, and Li Yuan. Act as you wish: Fine-grained control of motion diffusion model with hierarchical semantic graphs. *Advances in Neural Information Processing Systems*, 36:15497–15518, 2023. 2

[14] Tao Ju, Qian-Yi Zhou, Michiel Van De Panne, Daniel Cohen-Or, and Ulrich Neumann. Reusable skinning templates using cage-based deformations. *ACM Transactions on Graphics (ToG)*, 27(5):1–10, 2008. 3

[15] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 6

[16] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *Proceedings of the International Conference on Machine Learning*, pages 2688–2697. PMLR, 10–15 Jul 2018. 2

[17] Tianshu Kuai, Akash Karthikeyan, Yash Kant, Ashkan Mirzaei, and Igor Gilitschenski. Camm: Building category-agnostic and animatable 3D models from monocular videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6587–6597, 2023. 3

[18] Sumith Kulal, Jiayuan Mao, Alex Aiken, and Jiajun Wu. Hierarchical motion understanding via motion programs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6568–6576, 2021. 1, 2

[19] Mei Chee Leong, Hui Li Tan, Haosong Zhang, Liyuan Li, Feng Lin, and Joo Hwee Lim. Joint learning on the hierarchy representation for fine-grained human action recognition. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 1059–1063. IEEE, 2021. 1, 2

[20] Jiaman Li, Ruben Villegas, Duygu Ceylan, Jimei Yang, Zhengfei Kuang, Hao Li, and Yajie Zhao. Task-generic hierarchical human motion prior using VAEs. In *2021 International Conference on 3D Vision (3DV)*, pages 771–781. IEEE, 2021. 2

[21] Yiming Liang, Tianhan Xu, and Yuta Kikuchi. Himor: Monocular deformable gaussian reconstruction with hierarchical motion representation. *arXiv preprint arXiv:2504.06210*, 2025. 2, 3, 9

[22] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 3

[23] MARK EJ Newman. Networks: An introduction, 2010. 3

[24] Jesús R Nieto and Antonio Susín. Cage based deformations: A survey. In *Deformation Models: Tracking, Animation and Applications*, pages 75–99. Springer, 2012. 3

[25] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 9, 10

[26] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, 2020. 2

[27] Dian Shao, Yue Zhao, Bo Dai, and Dahua Lin. FineGym: A hierarchical video dataset for fine-grained action understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2616–2625, 2020. 2

[28] Aliaksandr Siarohin, Oliver J Woodford, Jian Ren, Menglei Chai, and Sergey Tulyakov. Motion representations for articulated animation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13653–13662, 2021. 2

[29] Olga Sorkine. Laplacian mesh processing. *Eurographics (State of the Art Reports)*, 4(4):1, 2005. 3

[30] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116. Citeseer, 2007. 3, 6

[31] Wataru Takano, Hirotaka Imagawa, and Yoshihiko Nakamura. Prediction of human behaviors in the future through symbolic inference. In *IEEE International Conference on Robotics and Automation*, pages 1970–1975. IEEE, 2011. 1

[32] Lingling Tao and René Vidal. Moving poselets: A discriminative and interpretable skeletal motion representation for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 61–69, 2015. 2

[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 5

[34] Can Wang, Mingming He, Menglei Chai, Dongdong Chen, and Jing Liao. Mesh-guided neural implicit field editing. *arXiv preprint arXiv:2312.02157*, 2023. 3

[35] Qianqian Wang, Vickie Ye, Hang Gao, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4D reconstruction from a single video. *arXiv preprint arXiv:2407.13764*, 2024. 6

[36] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4D gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20310–20320, 2024. 6

[37] Jingwei Xu, Huazhe Xu, Bingbing Ni, Xiaokang Yang, Xiaolong Wang, and Trevor Darrell. Hierarchical style-based networks for motion synthesis. In *Proceedings of the European Conference on Computer Vision*, pages 178–194. Springer, 2020. 2

[38] Kaizhi Yang, Xiaoshuai Zhang, Zhiao Huang, Xuejin Chen, Zexiang Xu, and Hao Su. MovingParts: Motion-based 3D part discovery in dynamic radiance field. *arXiv preprint arXiv:2303.05703*, 2023. 2, 3

[39] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3D gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20331–20341, 2024. 6

[40] Wang Yifan, Noam Aigerman, Vladimir G Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3D deformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 75–83, 2020. 3

[41] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in Neural Information Processing Systems*, 31, 2018. 3

[42] Kai Yuan, Noor Sajid, Karl Friston, and Zhibin Li. Hierarchical generative modelling for autonomous robots. *Nature Machine Intelligence*, 5(12):1402–1414, 2023. 1

[43] Xinyu Zhang, Haonan Chang, Yuhan Liu, and Abdeslam Boularias. Motion blender gaussian splatting for dynamic reconstruction. *arXiv preprint arXiv:2503.09040*, 2025. 3, 9

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly outline the problem of hierarchical motion modeling across domains and introduce the proposed graph-based learning method. The paper delivers on its stated contributions by formalizing the hierarchy learning problem, presenting the proposed method, and validating it on 1D and 3D motion modeling tasks.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The limitations of the method are explicitly discussed in the conclusion section, noting challenges such as dependence on observed motion data, and the assumption of a single-parent hierarchy per motion element.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [NA]

   Justification: The paper is primarily algorithmic. While it formalizes the problem and provides mathematical formulations of the model and training procedures, it does not present new theoretical proofs requiring validation.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The method section specifies key model components, equations and settings for both 1D motion and 3D Gaussian splatting tasks. The training and implementation details are discussed in the supplemental material.

   Guidelines:

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

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

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

   Answer: [Yes]

   Justification: The code and data will be released upon publication. Supplemental material includes detailed information on datasets, model configurations, and instructions for running the experiments.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The supplementary material list training parameters (e.g., learning rates, optimizers, epochs), data sources, and evaluation procedures.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: In the 1D motion hierarchy reconstruction task, we conduct 1000 independent trials and report aggregate accuracy. This provides a statistically meaningful evaluation of the model's robustness and generalization in a controlled setting.

   Guidelines:

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

8. **Experiments compute resources**

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

Answer: [Yes]

Justification: The supplemental material specifies the hardware (NVIDIA RTX 3090) and approximate runtimes for different experiments, which are sufficient to estimate required computational resources.

Guidelines:

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

9. **Code of ethics**

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

Answer: [Yes]

Justification: The research does not involve human subjects, sensitive data, or applications with clear misuse risks. The contributions focus on improving motion modeling in computer vision and graphics, with no evident ethical violations.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

Answer: [Yes]

Justification: While the primary focus is technical, the conclusion briefly discusses the method's applicability and limitations.

Guidelines:

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

11. **Safeguards**

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

Answer: [NA]

Justification: The method and datasets used do not pose high misuse risks. The models are task-specific and do not involve generative models with sensitive outputs.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: The paper cites all external methods (e.g., as-rigid-as-possible), datasets (D-NeRF), and baseline models used in experiments. References to all these are included.

Guidelines:

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

13. **New assets**

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

Answer: [Yes]

Justification: We introduce a new motion hierarchy learning method and a set of reproducible synthetic motion scenarios for evaluation. The paper and the supplemental material provide detailed documentation of the algorithmic pipeline, model design, data generation process, and implementation setup. Code release will include usage instructions and configuration files to support external use.

Guidelines:

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

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

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

Answer: [NA]

Justification: The paper does not involve any human subject experiments or crowdsourcing components.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

Answer: [NA]

Justification: No human subject research was conducted, and no IRB review was necessary.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [NA]

Justification: LLMs are not used in the method or experiments. Any use of language models was only for text editing and not relevant to the scientific content.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.