Motion Hierarchical Gaussian for Dynamic Control in VR

Runze Fan, Jian Wu, Qixiang Ma, Zhikai Wen, and Lili Wang

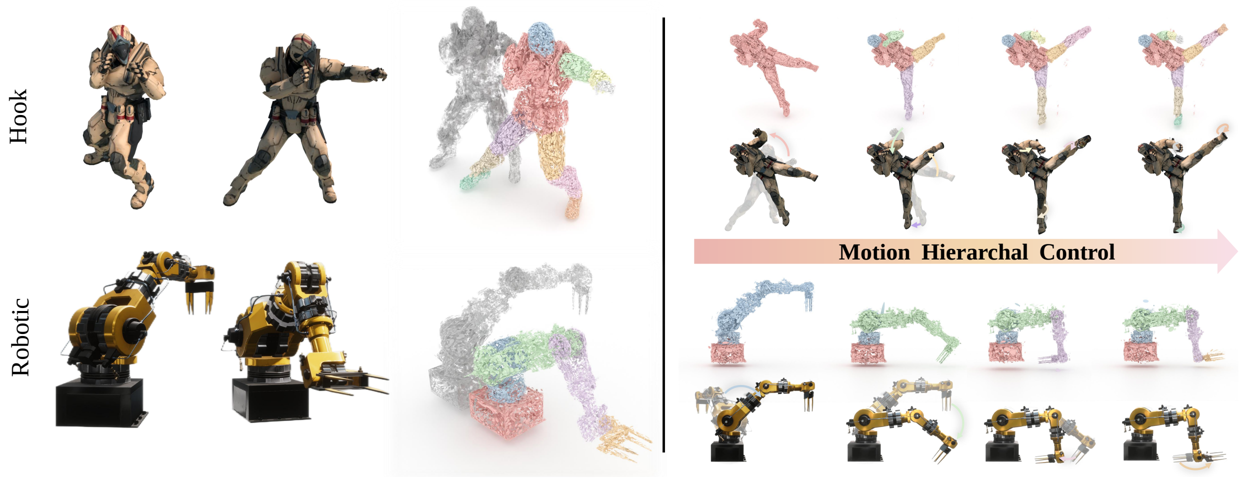


Fig. 1: Motion reconstruction and control results for *Robotic* and *Hook*. Motion hierarchical Gaussian is reconstructed based on the motion sequences in the dataset (left). The gray Gaussian indicates the starting motion and the colored Gaussian indicates the ending motion. Different colors represent the motion hierarchy of Gaussian. Motion control is executed in different layers of the hierarchy through a top-down approach (right), enabling the generation of novel, realistic, and structurally plausible motion sequences, even those that differ significantly from the original motion in the dataset.

Abstract— Intuitive motion control is essential for virtual reality, allowing users to manipulate objects naturally while receiving realistic and responsive visual feedback. 3D Gaussian splatting provides real-time, photorealistic scene rendering, making it promising for virtual reality applications. Still, it falls short in accurate motion control of dynamic objects due to its unstructured global motion representation and redundant motion learning. To address these problems, we propose a motion hierarchical Gaussian based dynamic control method. First, a motion hierarchical Gaussian representation is introduced and initialized with semantic and deformation information. Then a motion hierarchical decomposition method is proposed to optimize the local motion in the representation. The representation is next optimized by a local motion analysis based refinement method. We also design a set of motion control operations for the motion hierarchical Gaussian. Experimental results show that our method achieves high-precision motion reconstruction, accurate motion decomposition, real-time, intuitively and immersive VR motion control.

Index Terms—3D Gaussian Splatting, Motion Control, Real-Time Interaction.

1 Introduction

3D Gaussian splatting (3DGS) [1] has attracted significant attention in the field of virtual reality (VR) for its high rendering quality and real-time performance. Recent research focuses on enabling real-time interaction based on this Gaussian representation. Due to the explicit nature of 3DGS, simply scene geometry editing, such as object removal, can already be handled effectively [2–4]. However, more complex interactions, such as motion control and dynamic manipulation, still need to be studied. One idea is to perform physical simulations to model the dynamics. VR-GS [5] reconstructs mesh for Gaussians, and uses the mesh-based XPBD simulation to drive Gaussian motion. Similarly, VR-Doh [6] applies a grid-based MPM simulation, and Gaussians are deformed based on the simulated grid velocities. While VR-GS and VR-Doh enable immersive real-time VR interactions, it relies on indirect control of Gaussian motion via intermediate mesh or grid representations. PhysGS [7] directly simulates the physical dynamics of Gaussian using MPM, enabling accurate motion control. However, as it performs simulations on individual Gaussian, the efficiency degrades with the increasing number of Gaussians.

Deformation-field-based dynamic Gaussians [8–10] provide a promising direction for motion control. Several approaches attempt to achieve complex motion control by learning motion patterns from time-dependent datasets and enabling user interaction with the learned motions. Control Point-based method [11] models and controls the Gaussian motion using a set of learnable control points. Although the number of these control points is significantly smaller than that of Gaussians, it is still relatively large for VR (≈ 500). Global motion-based methods [8, 12, 13] model the motion of each Gaussian via the deformation field. However, since they manipulate the global motion of individual Gaussian, the resulting control is coarse and inflexible. Motion-basis-based methods [14–17] employ a one- or two-level coarse-to-fine manner for motion reconstruction and, lacking a Gaussian hierarchy or explicit motion–Gaussian correspondences, only achieve a coarse motion decomposition and are incapable of enabling motion control. Thus, to enable real-time intuitive motion control in VR, two

• *Lili Wang is with State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China; Peng Cheng Laboratory, Shenzhen, China. Lili Wang is the corresponding author. E-mail: wanglily@buaa.edu.cn.*
• *Runze Fan, Jian Wu, Qixiang Ma, and Zhikai Wen is with State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China. E-mail: by2106131@buaa.edu.cn, lanayawj@buaa.edu.cn, sycamore_ma@buaa.edu.cn, wenzhikai@buaa.edu.cn.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

problems must be addressed: 1) Unstructured global motion representation. Global motion representations modeled by the deformation field lack hierarchical structure, making fine-grained motion control difficult and un-intuitive. 2) Redundant motion learning. Learning independent motion parameters for each Gaussian leads to redundancy when modeling rigid or quasi-rigid objects. Mathematically, rigid motions in 3D are represented by elements of the Lie group $SE(3)$, each consisting of a rotation $\mathbf{R} \in SO(3)$ and a translation $\mathbf{T} \in \mathbb{R}^3$. The group $SE(3)$ is closed under composition, meaning that complex motions can be expressed as compositions of motion bases, each still representing a valid local motion.

In this paper, we propose a motion hierarchical Gaussian based dynamic control method in VR. We introduce a motion hierarchical Gaussian representation which is a hierarchical motion tree. In this tree, each node represents a set of Gaussians and its associated local motion, and each edge encodes the relationship between local motion at different hierarchy layers. The global motion of each Gaussian is recursively computed by propagating local motion from the root node down to its corresponding node in the tree. We propose a semantic and deformation based initialization method to construct this tree, which first initializes the motion layer (nodes) and then initialize the motion relationship (edges). To generate the local motion, we propose a motion hierarchical decomposition method based on consistency and orthogonality regularization. To refine the motion hierarchical Gaussian representation, we propose a local motion analysis based method, which optimizes the motion hierarchy by dynamically merging, removing, and splitting nodes, reassigning Gaussians, adjusting edge types, and updating local motion. We design a set of motion control operations for effectively interacting with the motion hierarchical Gaussian. We also conduct a user study to validate the effectiveness of our method. The results show the motion hierarchical Gaussian improves VR-specific interactions by enabling users to manipulate dynamic objects with high fidelity and real-time responsiveness, highlighting its practical advantages for VR dynamic control. Fig.1 visualizes the results of our method.

In summary, the contributions of this paper are as follows:

- We propose a new motion hierarchical Gaussian representation, which encodes the motion relation and local motion of each component of objects in the scene.
- We propose a consistency and orthogonality regularization based motion hierarchical decomposition method to generate the local motion of our representation.
- We present a local motion analysis based refinement method, which iteratively optimizes the nodes, edges and local motion.

2 Related Work

In this section, we review three types of researches related to our work: dynamic scenes reconstruction, 3D deformation editing, and hierarchical scene representation. For a more comprehensive overview of prior works, we recommend readers refer to these reviews [18, 19].

2.1 Dynamic scene reconstruction

With a given set of photos, dynamic scene reconstruction methods recover both the geometry and appearance of scenes that involve temporal changes. As representative neural representations, Neural Radiance Fields (NeRF) [20], and 3D Gaussian Splatting [1] demonstrate high photorealism in rendering, making them widely adopted for modeling complex scenes.

NeRF-based methods [21–31] introduced the deformation field to the radiance field and have shown impressive scene reconstruction and novel view synthesis results for dynamic scenes. However, these methods are time-consuming and unable to support real-time rendering and interaction in VR. Although some subsequent works [32–34] used HexPlane, K-Plane, and octree to accelerate computation, they still pose challenges to achieve real-time rendering.

Many concurrent works have adapted 3DGS to dynamic scenes due to high performance. Some methods extend the 3DGS with time-variant Gaussian features [35, 36]. Some methods decouple the deformation from the static Gaussian with the temporal conditioned deformation field [8, 9, 12]. To further improve the rendering performance, Fov-GS [37] introduces foveated rendering into 3DGS. These methods model the global deformation of the Gaussian, but directly controlling such global deformation may lead to unrealistic results. A more plausible solution is to control local motions instead. In this paper, we propose a motion hierarchical Gaussian representation that uses local motion rather than the deformation field to reconstruct the dynamic scene.

2.2 3D Deformation Editing

Existing NeRF-based works [38–46] aim to edit the geometry while preserving the details during the deformation. Due to its implicit nature, NeRF-based 3D deformation editing is not easy to control.

Thanks to the explicit nature of 3DGS, intuitive and direct geometry editing, such as deletion, rotation, and removing, can be easily performed with Gaussian semantic segmentation [3, 4, 47, 48]. However, these methods treat the Gaussians of the object as a whole, where operations are performed on the entire Gaussians rather than on individual parts. To enable flexible deformation and editing over each Gaussian, PhysGS [7] directly computing the deformation of each Gaussian through physical simulation. However, it suffers from high computational time. To improve simulation efficiency, various mesh-based Gaussian deformation and editing methods have been proposed [5, 49, 50]. These methods bind Gaussians to meshes, simulating the deformation of the mesh and driving the deformation of the Gaussians through the mesh's transformation. However, the simulation pipelines of these methods are relatively complex, and the Gaussian motion is controlled indirectly. To further improve simulation and rendering performance, VR-Doh [6] bind Gaussian to grid, simulating the grid deformation and driving the Gaussian motion by the grid velocity.

SC-GS [11] models and edits the Gaussian deformation using a set of sparse learnable control points. Due to the lack of accurate topology information between these control points and the large number of them, precise control and editing of deformation through these control points remains challenging. In this paper, we propose a motion hierarchical Gaussian method that directly controls and edits the motion of each Gaussian to achieve efficient, interactive, real-time motion control in VR.

2.3 Hierarchical Scene Representation

Hierarchy has long been fundamental in traditional scene representations, such as point cloud, mesh, and voxels [51–58]. With the advent of neural scene representations, hierarchy has been adopted to improve reconstruction quality and rendering efficiency [59–61].

Recent Gaussian-based methods provide an explicit, compact representation that naturally supports hierarchical organization. Examples include Hier-GS for large static scenes [62], HiSplat for coarse-to-fine static reconstruction [63], DualGS for two-level joint-skin modeling of humans [64], and TGSH for multi-level 4D Gaussians in volumetric video [16]. However, these methods focus on either static or human scenes, and none decompose motion at the Gaussian level, limiting their ability for 3D deformation editing.

Motion-basis-based methods [14, 15] decomposes motions into a single-level set of basis. HiMoR [17] further employs a fixed two-level coarse-to-fine hierarchy, enabling basic motion decomposition, but constructing the hierarchy only for motion. Because no corresponding Gaussian hierarchy or explicit motion–Gaussian correspondence is built, motion hierarchy become entangled, making hierarchy both difficult to interpret (editing one motion node propagates unintended deformation to many Gaussians) and unsuitable for fine motion control (the correspondence between motion nodes and total motions is ambiguous), ultimately hindering effective motion editing. Moreover, fixed two-level hierarchy is insufficient to model the cascaded, multi-level complexity of real-world motions. In this paper, we propose a motion hierarchical Gaussian representation which incorporates cascaded multi-level motion decomposition with corresponding motion and Gaussians hierarchy. This representation yields semantically interpretable motion and Gaussian nodes, and enables fine-grained, plausible motion editing.

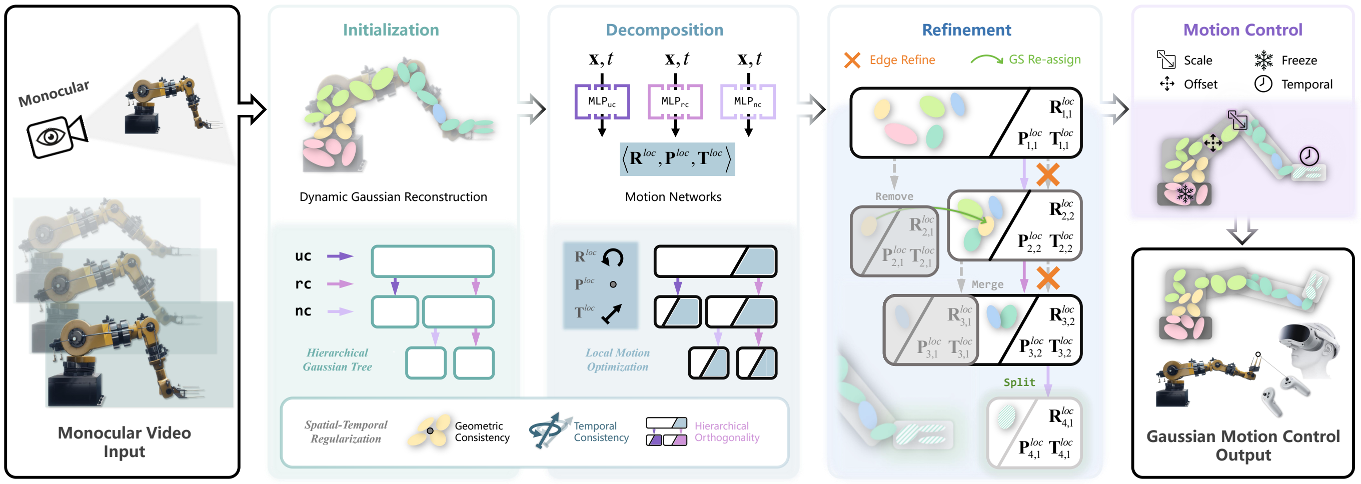


Fig. 2: The pipeline of our motion hierarchical Gaussian based dynamic control method.

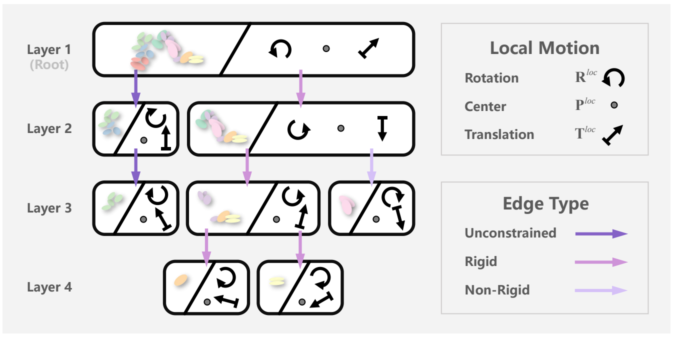


Fig. 3: Motion hierarchical Gaussian representation.

3 Method

Given a monocular video of a dynamic scene as input, we propose the motion hierarchical Gaussian representation (Sec. 3.1) that accurately reconstructs the dynamic scene and provides real-time, interactive multilayer motion control in VR. As shown in Fig. 3, we construct the motion hierarchical Gaussian with 3 steps: 1) semantic and deformation based initialization (Sec. 3.2); 2) motion hierarchical decomposition (Sec. 3.3); 3) local motion analysis based refinement (Sec. 3.4). After this, we leverage the explicit geometric representation (Gaussian) and decomposed local motion (motion hierarchy) reconstructed by the above methods to achieve efficient and intuitive Gaussian motion control (Sec. 3.5).

3.1 Motion Hierarchical Gaussian Representation

To decompose global motion into interpretable and controllable local motion, inspired by Lie groups and algebra, we propose a representation for modeling complex motion, **Motion Hierarchical Gaussian**, which achieves accuracy and intuitiveness through a tree structured motion hierarchy. The tree contains **Nodes** and **Edges**.

Node contains two components, Gaussian group with the same or similar local motion, and the corresponding local motion. We define **local motion** as a triplet $\langle \mathbf{R}^{loc}, \mathbf{P}^{loc}, \mathbf{T}^{loc} \rangle$, where $\mathbf{R}^{loc}$, $\mathbf{P}^{loc}$ and $\mathbf{T}^{loc}$ represents the rotation, its rotation center, and the original translation, receptively. We compute $\mathbf{T}^{loc}$ based on Lie group. The total translation $\mathbf{T}$ of Gaussian group consists of two parts: the original translation $\mathbf{T}^{loc}$ and the rotation-induced translation $\mathbf{T}^{rot}$. When a Gaussian rotates around a center $\mathbf{P}^{loc}$, the rotation $\mathbf{R}^{loc}$ induces an additional translation $\mathbf{T}^{rot} = (\mathbf{I} - \mathbf{R}^{loc})\mathbf{P}^{loc}$. The original $\mathbf{T}^{loc}$ is then computed as $\mathbf{T}^{loc} = \mathbf{T} - (\mathbf{I} - \mathbf{R}^{loc})\mathbf{P}^{loc}$. This disentanglement preserves the natural coupling between rotation and translation, leading to a more physically consistent and interpretable motion representation.

Edge represents the local motion relationship between nodes of different layers. There are three types of the local motion relationship: 1) **unconstrained uc**: Child motion is not affected by parent. 2) **rigid constraint rc**: Child motion is strictly dependent on parent, maintaining rigid consistency. 3) **non-rigid constraint nc**: Child motion is loosely dependent on parent, i.e., a perturbation is introduced to rigid motion.

For the node with **uc** edge connecting its parent, the global motion of each Gaussian in the Gaussian group is determined by Eq. 1, i.e., the global motion equals to the local motion.

$$\begin{cases} \mathbf{x}(t) = \mathbf{P}^{loc}(t) + \mathbf{R}^{loc}(t)\left(\mathbf{x} - \mathbf{P}^{loc}(t)\right) + \mathbf{T}^{loc}(t) \\ \mathbf{r}(t) = \mathbf{R}^{loc}(t)\mathbf{r} \end{cases} \tag{1}$$

where $\mathbf{x}$ and $\mathbf{r}$ are the 3D center and rotation of the Gaussian in canonical space, and t is the current time.

For the node with **rc** edge connecting its parent, the global motion of Gaussian is factorized into a chain of local motions due to the associative property in $SE(3)$. For a Gaussian in node N_l with local motion $\left\langle \mathbf{R}^{loc}_{N_l}, \mathbf{P}^{loc}_{N_l}, \mathbf{T}^{loc}_{N_l} \right\rangle$ at layer l, the center $\mathbf{x}_{N_l}$ is determined with Eq.2

$$\begin{cases} \mathbf{x}_{N_l}(t) = \mathbf{P}_{N_l}(t) + \mathbf{R}_{N_l}(t)\left(\mathbf{x}_{N_{l-1}}(t) - \mathbf{P}_{N_l}(t)\right) + \mathbf{T}_{N_l}(t) \\ \mathbf{x}_{N_0}(t) = \mathbf{P}_{N_0}(t) + \mathbf{R}_{N_0}(t)\left(\mathbf{x} - \mathbf{P}_{N_0}(t)\right) + \mathbf{T}_{N_0}(t) \end{cases} \tag{2}$$

where $\mathbf{x}_{N_{l-1}}$ is the transformed Gaussian center with the local motion of its parent node N_{l-1}, $\mathbf{x}_{N_0}$ is the transformed Gaussian center with the local motion of the root node N_0. The rotation $\mathbf{r}_{N_n}$ is determined with Eq.3

$$\begin{cases} \mathbf{r}_{N_n}(t) = \mathbf{R}^{loc}_{N_l}(t)\mathbf{r}_{N_{l-1}}(t) \\ \mathbf{r}_{N_0}(t) = \mathbf{R}^{loc}_{N_0}(t)\mathbf{r} \end{cases} \tag{3}$$

where $\mathbf{r}_{N_{l-1}}$ and $\mathbf{r}_{N_0}$ are the transformed Gaussian rotation with the local motion of N_{l-1} and N_0.

For the node with **nc** edge connecting its parent, the global motion is computed as the composition of the parent's motion and the child's local motion, with an additional non-rigid perturbation to account for incomplete rigidity. We approximate this non-rigid perturbation as a small offset of the rotation center $\Delta\mathbf{P}$ only, while keeping the local rotation and translation identical to the rigid motion. For a Gaussian in node N_l, the final local motion is therefore formulated as $\left\langle \mathbf{R}^{loc}_{N_l}, \mathbf{P}^{loc}_{N_l} + \Delta\mathbf{P}^{loc}_{N_l}, \mathbf{T}^{loc}_{N_l} \right\rangle$. The global motion can then be calculated using Eq.2 and 3.

3.2 Semantic and Deformation Based Initialization

We propose a semantic and deformation based clustering method to initialize the motion hierarchical Gaussian. Taking the monocular video of a dynamic scene as input, we first generate L groups of ID consistent 2D semantic masks ranging from coarse to fine with level-specific segmentation granularities, based on SAM and deva3 [65, 66], following prior works [2, 67]. Then, a set of Gaussians, corresponding deformation field and ID properties are trained. The deformed 3D Gaussian and multi-granularity ID properties are trained supervised by RGB images and corresponding granularity 2D masks, following [2, 8, 12]. For each granularity level $l = 1, \dots, L$, an independent ID property ID_{N_l} of dimension 2^{5+l} is trained, while all levels share the common deformation field and Gaussian.

After this, the nodes of our motion hierarchical Gaussian tree with $(L+1)$ layers are initialized using a top-down recursive clustering strategy. We begin by constructing the node of the root layer (0 layer). All the initialized 3D Gaussian are grouped in the root node. The local motion of this node is initialized with $\mathbf{0}$. The Gaussians of the nodes in l layer ($l > 0$) are grouped based on the semantic-deformation feature F_l. This feature contains the deformation feature and ID property ID_{N_l}. The deformation feature is generated by uniformly sampling the deformation field with $K_l = max(2^{5+l}, n)$ time steps, where n denotes the number of the training images. Then, the Gaussians of the nodes in $l-1$-th layer are assigned to the nodes in l-th layer by using HDBSCAN clustering [68] with F_l. The local motions of these nodes are also initialized with $\mathbf{0}$.

At last, the edges are initialized via deformation similarity analysis, since the initial Gaussians contain only deformation information but lack local motion cues, and deformation can be regarded as a coarse description for motion. Deformation similarity is introduced to evaluate the similarity of global motion between parent and child nodes, which includes positional similarity and rotational similarity. For a given parent-child node pair, positional similarity S_x is defined with Eq. 4

$$S_x = \frac{\sum_t \|\delta x^{\mathcal{P}}(t) - \delta x^{\mathcal{C}}(t)\|}{Var(\delta x^{\mathcal{P}})\, Var(\delta x^{\mathcal{P}})} \tag{4}$$

where $\delta x^{\mathcal{P}}$ and $\delta x^{\mathcal{C}}$ are the mean positional offsets of all Gaussians in the parent and child nodes at time t, and $Var(\delta x)$ denotes the variance of those offsets over t. High S_x indicates independent (unconstrained) motion, while low S_x signifies coupled motion. Rotational similarity S_r is defined with Eq. 5

$$S_r = Var\left(\left\|\log\left((\delta r^{\mathcal{P}}(t))^{-1}\delta r^{\mathcal{C}}(t)\right)\right\|_F\right) \tag{5}$$

where $\delta r^{\mathcal{P}}$ and $\delta r^{\mathcal{C}}$ are the mean rotational offsets. The logarithm maps the relative rotation into the Lie algebra $so(3)$, and Frobenius norm $\|\|_F$ is then used to to quantify the magnitude of the relative rotation. Low S_r indicates the child's rotation follows the parent's rigid transform, whereas high S_r reveals non-rigid. We then assign each edge type based on a similarity test. First, S_x is evaluated: if it exceeds a threshold, the edge is assigned as **uc** (unconstrained). Otherwise, S_r is further evaluated: if S_r is below a second threshold, the edge is assigned as **rc** (rigid constraint); otherwise, it is assigned as **nc** (non-rigid constraint).

3.3 Motion Hierarchical Decomposition

In this section, we compute the local motion of each node by decomposing the global motion based on the hierarchical motion tree. Three types of motion networks are proposed to predict the local motion for different edge types and are optimized with three regularization terms.

Motion network definition. We design three types of motion networks, $MLP_{\mathbf{uc}}$, $MLP_{\mathbf{rc}}$ and $MLP_{\mathbf{nc}}$, to predict the local motion parameters $\langle \mathbf{R}^{loc}, \mathbf{P}^{loc}, \mathbf{T}^{loc} \rangle$ for Gaussians in nodes with **uc**, **rc** and **nc** edges, respectively.

$MLP_{\mathbf{uc}}$: The local motion equals to the global motion due to child motion is not affected by parent, so all motion parameters are learned freely from the canonical space. The network takes as input the Gaussian center $\mathbf{x}$ in canonical space and the time t, and directly predicts all three local motion parameters $\langle \mathbf{R}^{loc}, \mathbf{P}^{loc}, \mathbf{T}^{loc} \rangle$.

$MLP_{\mathbf{rc}}$: Child local motion maintains rigid consistency with parent's. The network takes as input the transformed Gaussian center $\mathbf{x}^{t\mathcal{P}}$ in parent coordinate space and t, and predicts $\mathbf{R}^{loc}$, $\mathbf{P}^{C}$, and $\mathbf{T}^{loc}$. $\mathbf{x}^{t\mathcal{P}}$ is calculated with Eq.2 ($\mathbf{x}_{N_{l-1}}$). $\mathbf{P}^{C}$ is the rotation center in canonical space. The final local rotation center $\mathbf{P}^{loc}$ is calculated with Eq.2 by substituting $\mathbf{P}^{C}$ as $\mathbf{x}$ in the base recurrence and then propagating the recurrence up the hierarchy. The resulting $\mathbf{x}_{N_l}$ is the local rotation center $\mathbf{P}^{loc}$ of that Gaussian at layer l.

$MLP_{\mathbf{nc}}$: Child motion is non-rigid to the parent's with a perturbation capturing deviations from rigid motion. The network again takes as input $\mathbf{x}^{t\mathcal{P}}$ and t but predicts $\mathbf{R}^{loc}$, $\mathbf{P}^{C}$, $\Delta\mathbf{P}$ and $\mathbf{T}^{loc}$. The final local rotation center $\mathbf{P}^{loc}$ is calculated as $\mathbf{P}^{loc} = \mathbf{P}^{C} + \Delta\mathbf{P}$.

For each node in the hierarchical motion tree, all associated Gaussians share the same MLP. Similar to the DGS [8], we supervise these MLPs using 2D ground-truth images with RGB reconstruction loss. To ensure structured and coherent motion decomposition by enforcing spatial geometry consistency, temporal motion consistency and hierarchical motion orthogonality, we introduce three regularization terms.

Spatial Geometry Consistency Regularization. For Gaussians within the same node which subjects to rigid or non-rigid constraints, their $\mathbf{P}^{C}$ should be as similar as possible. We introduce a spatial geometry consistency regularization term $\mathcal{L}_{S-consis}$ that maximize the pairwise consistency of $\mathbf{P}^{C}$ with Eq.6.

$$\mathcal{L}_{S-consis} = \sum_{i,j} \left\|\mathbf{P}_i^{C} - \mathbf{P}_j^{C}\right\|^2 \tag{6}$$

where $\mathbf{P}_i^{C}$ and $\mathbf{P}_j^{C}$ denote the rotation center in the canonical space for different Gaussians within the same node.

Temporal Motion Consistency Regularization. Gaussians within the same node are expected to undergo consistent motion. To enforce this consistency, we introduce a temporal motion consistency regularization term $\mathcal{L}_{T-consis}$ that minimizes the intra-node variance of predicted local motion parameters with Eq.7.

$$\mathcal{L}_{T-consis} = \frac{1}{n} \sum_{M \in \{\mathbf{R}^{loc}, \mathbf{T}^{loc}\}} \sum_{i=1}^{n} \left\|M_i - \bar{M}\right\|^2 \tag{7}$$

where $\bar{M}$ represents the average value of the predicted local motion parameter M across all n Gaussians in the node.

We employ two separate consistency regularization terms because the spatial geometry consistency regularization operates on the time-invariant canonical centers $\mathbf{P}_i^{C}$, and is therefore applied only once at the end of each training epoch. The temporal motion consistency regularization is time-dependent and applied for each time step.

Hierarchical Orthogonality Regularization. Local motion between parent and child nodes are encouraged to be sufficiently distinct. Hierarchical orthogonality encourages a more compact motion hierarchy by reducing the number of layers, thereby mitigating redundancy in local motion decomposition. Therefore, we introduce hierarchical orthogonality regularization that penalizes rotational similarity between parent and child nodes. We define this regularization term $\mathcal{L}_{H-orth}$ using the Frobenius norm in $SO(3)$ based on Eq.8.

$$\mathcal{L}_{H-orth} = \exp\left(-\left\|\log\left(\left(\bar{\mathbf{R}}^{\mathcal{P}}\right)^{-1}\bar{\mathbf{R}}^{\mathcal{C}}\right)\right\|_F\right) \tag{8}$$

where $\bar{\mathbf{R}}^{\mathcal{P}}$ and $\bar{\mathbf{R}}^{\mathcal{C}}$ denote the average local rotation predicted for the parent and child nodes, respectively.

The overall loss $\mathcal{L}$ is formulated with Eq. 9.

$$\mathcal{L} = \mathcal{L}_c + \alpha\mathcal{L}_{S-consis} + \beta\mathcal{L}_{T-consis} + \gamma\mathcal{L}_{H-orth} \tag{9}$$

where $\mathcal{L}_c$ is the RGB reconstruction loss combining L1 loss with the D-SSIM term from [1]. We set β and γ as 0.1 for all nodes. For nodes connected to their parents via **nc** or **rc** edges, we set α to 0.05, and for nodes connected via **uc** edges, α is set to 0.

3.4 Local Motion Analysis based Refinement

To address inaccuracies in nodes and edges of the motion hierarchical Gaussian obtained from previous two steps, we propose a refinement method based on local motion analysis. We define the **local motion**

similarity between two nodes with a triplet $\langle S_{\mathbf{R}}, S_{\mathbf{P}}, S_{\mathbf{T}} \rangle$, where $S_{\mathbf{P}}$ and $S_{\mathbf{T}}$ measure the similarity of rotation center and local translations, computed following Eq.4, and $S_{\mathbf{R}}$ measures the rotational similarity, computed as in Eq. 5. For each node, we define the **local motion discrepancy** as a 5-tuple $\langle m_{\mathbf{R}}, m_{\mathbf{T}}, v_{\mathbf{R}}, v_{\mathbf{P}}, v_{\mathbf{T}} \rangle$, representing the average of $\mathbf{R}^{loc}$ and $\mathbf{T}^{loc}$, and variance of $\mathbf{R}^{loc}$, $\mathbf{P}^{loc}$ and $\mathbf{T}^{loc}$ across all Gaussians and all time steps.

In node refinement, we first address *over-clustering*. Two nodes in the same layer with their local motion similarity below threshold θ_{merge} and share a common parent and are connected to it via either **nc** or **rc** edges or are connected to **uc** edges of different parents, we perform the ***merge*** operation. Merging is conducted iteratively, following a size-based strategy in which the node with fewer Gaussians is preferentially merged into the node with more. Additionally, we ***remove*** nodes whose local motion discrepancy falls below threshold θ_{remove}, indicating that the node do not contribute meaningful motion decomposition. To resolve *under-clustering*, we ***split*** a node into two nodes if its local motion discrepancy exceeds threshold θ_{split}. For nodes connected to their parents via **uc** edges, the split sub-nodes remain at the same layer and are connected to the parents with new **uc** edges. Otherwise, the sub-nodes are assigned to the next layer and connected to the parents via **nc** edges. To address *wrong-clustering*, we perform Gaussian-level ***reassignment***. We first identify nodes whose local motion discrepancy exceeds threshold θ_{reassign}. For each such node, we compute the deviation of each Gaussian's local motion from the node's mean local motion, rank all Gaussians by this deviation, and select the top 10% as outliers for reassignment. For nodes connected to their parent via **uc** edges, we search for candidate nodes across all nodes at the same layer that are also linked by **uc** edges. For nodes connected via **rc** or **nc** edges, the search is restricted to nodes under the same parent. Each outlier Gaussian is then reassigned to the most similar candidate node based on local motion similarity. If no better node is found, the Gaussian remains with its original node.

In edge refinement, we aggregate $\mathbf{R}^{loc}$ and $\mathbf{T}^{loc}$ for each node, and perform PCA on them. We use the explained variance ratio EVR of the top three principal components to quantify **spatial freedom** of motion. A Low EVR indicates that top-three principal components explain less eigenvalues, suggesting more unconstrained motion. In contrast, constrained motion exhibits a consistent pattern, with the top three components explaining most of the eigenvalues. We then extract the dominant principal component of the local motion at each time step, compute the cosine similarity between the principal directions of adjacent time steps and use the average similarity sim across all time steps to quantify the **temporal freedom**. A Low sim indicates unstable motion directions over time, i.e., unconstrained motion. While a high sim indicates stable, consistent motion directions, i.e., constrained motion. If both EVR and sim fall below thresholds $\theta_{\text{free-S}}$ and $\theta_{\text{free-T}}$, we set the edge type to **uc**. Otherwise, we backtrack the rotation center $\mathbf{P}^{loc}$ of the child node to canonical space with Eq.2 and calculate the variance of the backtracked rotation center as **axial instability**. For parent-child node pair with E_{rc}, the rotation axis P^{loc} of the child node should obey the motion of the parent nodes leading to low axial instability. Accordingly, if axial instability is below threshold $\theta_{\text{insta-axis}}$, we refine the edge type to **rc**. Otherwise, we set the edge type to **nc**.

3.5 Motion Control for Motion Hierarchical Gaussian

Based on the motion hierarchical Gaussian, we define five operations to enable flexible, intuitive, and physically plausible motion control in VR. These operations can be freely combined to produce diverse motion effects.

Motion Node Selection. Users first select a motion layer to determine the control granularity and then choose a specific node within the selected layer to control the local motion of the Gaussian belongs to it. **Freezing Local Motion**. User can freeze local motion by disabling its MLPs without affecting the local motion of other nodes. **Offsetting Local Motion**. User can adding 3D offsets to $\mathbf{R}^{loc}$, $\mathbf{P}^{loc}$ and $\mathbf{T}^{loc}$ to shift the local motion trajectory, control displacements, or alter motion directions. **Scaling Motion Magnitude**. By scaling the predicted $\mathbf{R}^{loc}$ and $\mathbf{T}^{loc}$, users can amplify or attenuate motion while maintaining physical plausibility. **Temporal Control**. User can modify the time input to the MLP of a selected node to decouple its motion from the global timeline, adjust the speed of local motion or introduce temporal delays, enabling effects like asynchronous movement or staggered activation. Editing the local motion of a node does not influence the local motion of its ancestor or children nodes, but it does impact the global motion of its children due to the hierarchy.

4 Experiment

4.1 Implementation

Datasets. We evaluate the reconstruction quality and segmentation quality of our method on D-NeRF [21], Hyper-NeRF [22] datasets and an additional Robotic dataset. The D-NeRF dataset and Hyper-NeRF dataset are monocular synthetic and real-world dataset, respectively. The Robotic dataset is also a monocular simulation dataset we created using Unity, designed to evaluate motion reconstruction and decomposition quality under diverse articulated motions. It contains six robotic arms with distinct geometries and textures, each exhibiting unique movement patterns such as rotation, extension, and grasping. For each scene, 200–300 high-resolution views (1920×1080) are rendered depending on the motion span. The camera viewpoints are sampled on a surrounding sphere following the strategy of D-NeRF. We will release the datasets with defined train/validation/test splits.

Implementation Details. We implemented our method on top of the original 3DGS [1] implementation. The deformation field used in Sec. 3.2 is the same as in DGS [8]. The structure of $MLP_{\mathbf{uc}}$, $MLP_{\mathbf{rc}}$ and $MLP_{\mathbf{nc}}$ are the same as the deformation field. The range of motion parameters is normalized for consistent thresholding: rotation $\mathbf{R} \in [0, 2\pi]$, position $\mathbf{P} \in [0, S]$, and translation $\mathbf{T} \in [0, T]$, where S denotes the object size and T the maximum absolute translation estimated from the initialized deformation field. Our refinement procedure follows a split–merge paradigm similar to classical clustering methods, such as K-Means, ISODATA, and HDBSCAN [68–70]. We define thresholds relative to the motion ranges, rather than in absolute units, to ensure scalability across different datasets. For edge initialization, thresholds for positional and rotational similarity (S_x, S_r) are set to $0.5T$ and $0.25\pi^2$, respectively. For refinement, merge operations are triggered when local motion similarity falls below 10% of the range, split operations are applied when local motion discrepancy exceeds 150% of the range, and remove operations are performed for nodes whose local motion discrepancy falls below 2% of the range. Reassignment is conducted when local motion deviation exceeds 100% of the range. These values are inspired by the general ranges used in ISODATA and HDBSCAN ($10\% - 50\%$ for merge, $120\% - 200\%$ for split, and $0.5\% - 3\%$ for remove), and were further validated empirically to provide stable performance across datasets. Since all thresholds are defined relative to the normalized motion ranges and follow a split–merge paradigm, our method is robust to moderate variations of these hyper-parameters. In practice, we observe that small changes in these thresholds do not significantly affect the final hierarchy or reconstruction quality. The thresholds for EVR and sim are set as 0.3 and -0.5, and for variance of P^C it is set as $0.01S^2$.

The initial Gaussian is trained for 40k iterations following DGS [8]. During the motion hierarchical decomposition, the motion networks are trained for 30k iterations. Motion hierarchical Gaussian refinement is performed 4 times, followed by motion networks optimization for 20k iterations. Gaussian-level reassignment is conducted every 5k iterations during motion networks optimization after each motion hierarchical refinement. Although our pipeline involves multiple training and refinement stages, these steps are performed entirely offline and do not affect runtime performance or VR interaction latency. After initialization, Gaussians are not dynamically created or removed. Each refinement iteration reuses the same optimization procedure, which keeps the implementation simple and consistent. The refinement converges within a small number of iterations (4 in all experiments). Once training is completed, the resulting representation supports real-time rendering and motion control without additional optimization. All experiments are conducted on an NVIDIA GeForce GTX 4090 GPU.

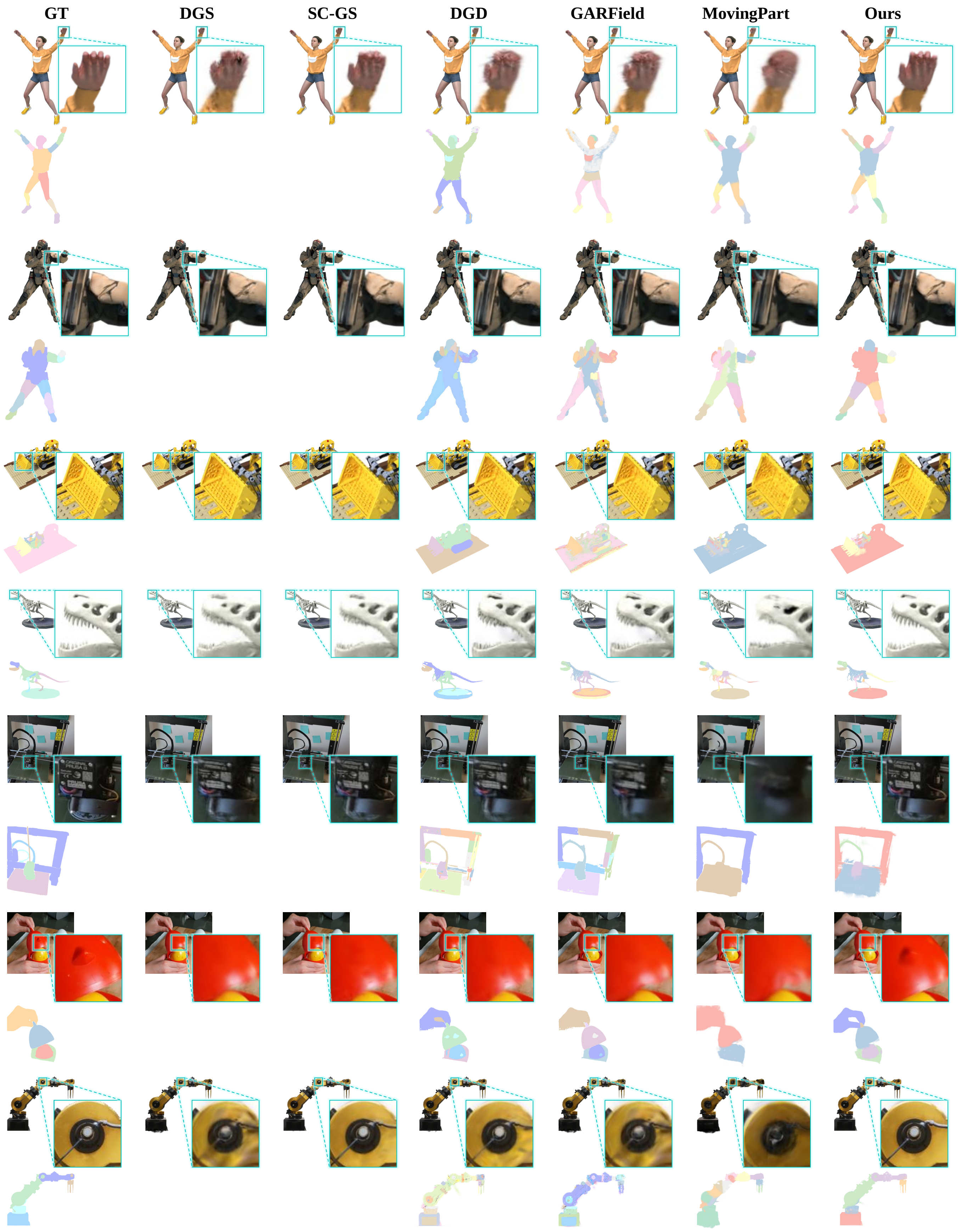


Fig. 4: Visual comparisons of reconstruction and segment quality between GT, the SOTA methods and our method on *Jumping*, *Hook*, *Lego*, *Trex*, *3DPrinter*, *Chicken* and *Robotic*.

Table 1: Reconstruction quantitative comparison on *Jumping*, *Hook*, *Lego*, *Trex*, *3DPrinter*, *Chicken* and *Robotic*.

Method	Jumping		Hook		Lego		Trex		3DPrinter		Chicken		Robotic	
	PSNR ↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓
DGS	38.34	0.0182	33.58	0.0195	32.95	0.0221	38.04	0.0089	28.39	0.2188	30.77	0.3899	27.99	0.0236
SC-GS	41.29	0.0124	34.46	0.0183	33.06	0.0291	40.24	0.0066	29.93	0.2105	31.41	0.3775	29.34	0.0183
DGD	35.23	0.0229	33.44	0.0198	32.24	0.0234	36.91	0.0105	27.51	0.2148	31.22	0.3828	29.95	0.0176
GARField	34.94	0.0243	31.69	0.0295	28.58	0.0464	35.47	0.0127	26.22	0.2682	29.61	0.4087	25.21	0.0332
MovingPart	32.44	0.0323	29.73	0.0298	31.14	0.0271	33.17	0.0154	22.01	0.4172	28.25	0.4225	25.81	0.0331
Ours	**41.39**	**0.0085**	**35.01**	**0.0156**	**37.44**	**0.0127**	**41.80**	**0.0058**	**30.19**	**0.2045**	**32.19**	**0.3596**	**33.63**	**0.0115**

4.2 Comparison

We compare our method with SOTA 3D Gaussian-based and NeRF-based methods, DGS [8], SC-GS [11], DGD [12], GARField [67] and MovingPart [13]. For GARField, which is designed for static scenes, we extend it with the deformation field to enable modeling of dynamic scenes. All baselines are compared under their intended input conditions. DGS, SC-GS and MovingPart take only the dynamic videos as input. While DGD, GARFiled and our method take the dynamic videos and semantic masks as input. Comparison metrics are selected according to each method's capabilities: reconstruction-only methods (DGS, SC-GS) are compared using reconstruction quality metrics, methods supporting segmentation (DGD, GARField, MovingPart, Ours) are compared using both reconstruction and segmentation metrics.

Reconstruction Quality. Fig. 4 shows the visual comparison between ground-truth, the results of SOTA methods and that of our method. Our results demonstrate higher similarity to the ground-truth, with clearer details for scenes in different datasets. This highlights the ability of our method to accurately model the local motion as well as hierarchical motions, both rigid and non-rigid. Table 1 shows the quantitative reconstruction quality comparison of the scenes in Fig.4. Table 2 shows the reconstruction quality comparison averaged over the D-NeRF and Hyper-NeRF datasets. The quantitative evaluation offers compelling evidence of the superior performance of our method.

Segmentation Quality. The segmentation ground-truth is based on the annotations provided by DGD. Since DGD performs semantic-based segmentation with relatively coarse granularity (e.g., in the *Jumping* scene, both the left and right hands are grouped into a single instance), we performed light manual refinements to produce more accurate and fine-grained annotations. For example, we separate the left and right hands into distinct segments. All methods are trained without access to ground-truth segmentation, and the ground-truth are used only for evaluation purposes. Specifically, for DGD, we use videos and segmentation masks generated by SAM [65] as supervision. For GARField, we also follow the original protocol and train the model using videos and semantic masks generated by SAM's automatic mask generator [65]. MovingPart is trained in a fully unsupervised manner, requiring no precomputed segmentation masks. For our method, we adopt the same strategy as DGD, using videos associated with multi-granularity segmentation masks for initialization.

Fig.4 visualizes the comparison of the segmentation results. Our method, DGD and GARField produce the finest-grained segmentation results directly, while MovingPart provides segmentation after group merging. Compared to SOTA methods, our method demonstrate higher similarity to the ground-truth, and significantly reduce both over-and under-segmentation, resulting in more accurate and reliable segmentation. This highlights the ability of our method in accurately modeling the hierarchical motions and the relationship between the local motions. We use mIoU in Tables 2 and 3 to quantitatively evaluate the segmentation quality. Our method achieves higher segmentation quality compared with the SOTA methods.

The last column in Table 2 shows the segmentation, hierarchy and motion control ability. DGS does not support segment and motion control. SC-GS only supports simple motion control with control-points. DGD only supports semantic segmentation. GARField only supports hierarchal semantic segment. MovingParts only supports hierarchal motion segment. Our method is the only to support all. Fig.5 visualize the comparison of the hierarchical Gaussian decomposition between our method, GARField and MovingPart. Compared with GARField and MovingPart, our method decomposes the motion step by step, reduces the over-segmented and under-segmented portions.

The hierarchical structure of our representation is essential for modeling structured local motions and enabling interactive motion control. A hypothetical single-layer variant, where all Gaussians are directly attached to the root, would collapse local motions into a global motion space, removing relative motion modeling and motion control, making such a variant infeasible for VR interaction. Moreover, several existing baselines, such as DGS and DGD, can be regarded as non-hierarchical motion representations, where motions are globally defined and independently learned per Gaussian. As demonstrated above, these formulations exhibit poorer reconstruction and segment quality compared to our hierarchical representation, highlighting the necessity of motion hierarchy.

Table 2: Comparison of reconstruction and segmentation quality, and segmentation, hierarchy, and motion control ability.

Method	D-NeRF			Hyper-NeRF			Seg	Hier	Ctrl
	PSNR↑	LPIPS↓	mIoU↑	PSNR↑	LPIPS↓	mIoU↑			
DGS	35.44	0.0184	-	30.58	0.2143	-	N	N	N
SC-GS	37.19	0.0160	-	32.68	0.2040	-	N	N	Y
DGD	35.68	0.0188	0.3078	32.36	0.2053	0.6402	Y	N	N
GARField	32.75	0.0254	0.3317	29.27	0.2316	0.6671	Y	Y	N
MovingPart	32.14	0.0247	0.3105	27.35	0.2879	0.5203	Y	Y	N
Ours	**41.23**	**0.0085**	**0.7694**	**34.96**	**0.1904**	**0.7069**	Y	Y	Y

Table 3: Segmentation quantitative comparison on different scenes.

mIoU↑	Jumping	Hook	Lego	Trex	3DPrinter	Chicken	Robotic
DGD	0.3352	0.4867	0.3266	0.2695	0.5146	0.7959	0.5875
GARField	0.3795	0.4583	0.3363	0.2809	0.5151	0.8363	0.5583
MovingPart	0.3358	0.4920	0.3081	0.2879	0.4142	0.6366	0.3689
Ours	**0.8775**	**0.7766**	**0.5898**	**0.8502**	**0.5361**	**0.8946**	**0.8671**

Fig.6 visualizes the comparison of the hierarchical motion tree before and after refinement. To better illustrate the hierarchical structure, we adopt an ID-inheritance rule where the first child of a parent node retains the parent's ID, while subsequent children receive incremented IDs. Our refinement algorithm effectively resolves over-clustering, under-clustering, and wrong-clustering of Gaussian in the initial hierarchy (e.g., the arm parts in *Robotic*, the hands in *Hook* and *Jumping*, and the tail vertebrae and vehicle cabins in *Trex* and *Lego*). Unlike GARField and MovingPart, which rely solely on either semantic or deformation cues, our initialization method jointly leverages both, leading to distinct hierarchies. Moreover, the hierarchy is refined by motion analysis, enabling precise motion decomposition (e.g., the boxer's right hand and body in *Hook*, or the vehicle body and cabin in *Lego*). Although different initialization strategies may yield different initial hierarchies, our method iteratively refines the hierarchy via motion decomposition and local motion analysis, leading to highly consistent final hierarchies that are largely invariant to the semantic prior (see supplementary for details).

Table 4: Performance(ms) and Gaussian number on different scenes.

Process	Jumping	Hook	Lego	Trex	3DPrinter	Chicken	Robotic
LocMPred	2.57	4.23	9.18	7.45	25.17	25.42	24.98
GloMCal	0.53	0.87	1.35	1.29	2.71	2.83	2.68
Render	0.41	0.93	1.47	1.21	2.22	2.47	2.09
Total	3.51	6.03	12.00	9.95	30.1	30.72	29.75
GS_Num	31K	58K	122K	92K	339K	340K	318K

Table 4 reports the runtime breakdown and model size. During rendering, our pipeline includes three stages: (1) local motion prediction

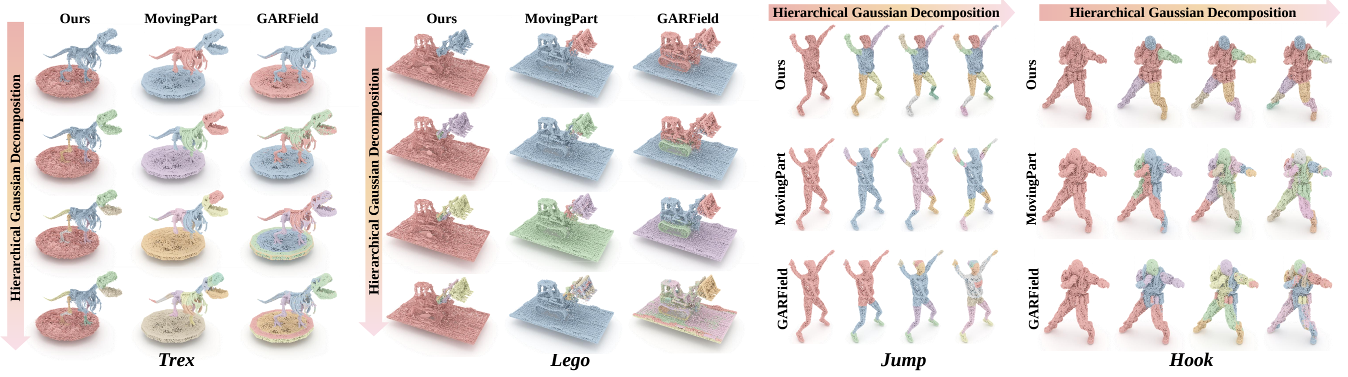


Fig. 5: Visual comparison of hierarchical Gaussian decomposition. Our method directly renders Gaussians as ellipsoids. GARField and MovingPart first sample the radiance field to generate point clouds, which are then rendered to visualize hierarchical results.

for each node in the hierarchical motion tree (LocMPred), (2) global motion computation based on Eq. 1–3 (GloMCal), and (3) Gaussian rendering (Render). Our method achieves real-time performance, with rendering speeds ranging from 32 to 284 fps across different scenes. We also report the number of Gaussians (GS_Num) per scene. Although the number of Gaussians varies by an order of magnitude, the runtime of GloMCal remains relatively stable, as the cascaded global motion computation consists only of simple mathematical operations.

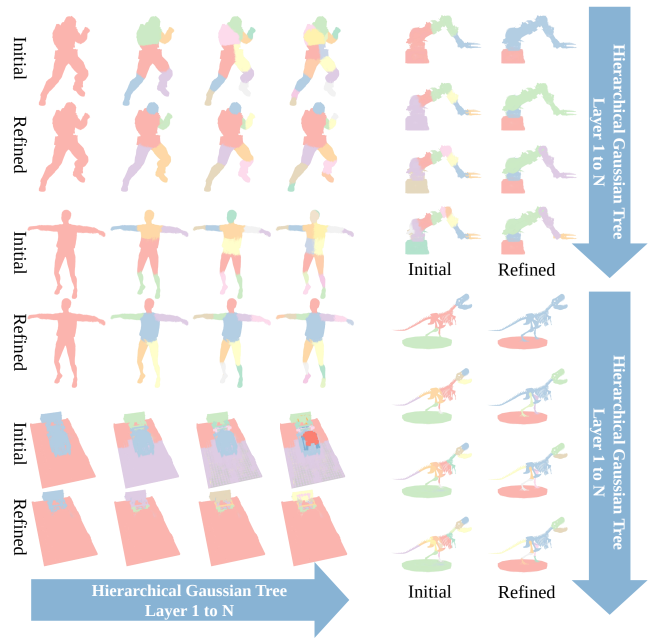


Fig. 6: Visual comparison of hierarchical Gaussian tree before and after local motion analysis based refinement.

4.3 Ablation Studies

We conducted ablation studies to validate the effectiveness of our proposed component, including regularization terms and motion hierarchal Gaussian refinement. We conduct experiments on the D-NeRF, Hyper-NeRF datasets and our Robotic datasets. The qualitative and quantitative results are reported in Fig.7 and Table 5.

Regularization. We first examine the effects of the three proposed regularization terms. Our method achieves the best performance when all regularization terms are enabled. As shown in Fig. 7, removing spatial geometry consistency (w/o R1) leads to mis-segmentation at the junctions between different parts of the *Robotic*, as well as incorrect reconstruction of local motions. Disabling temporal motion consistency (w/o R2) causes large motion deviations among Gaussians

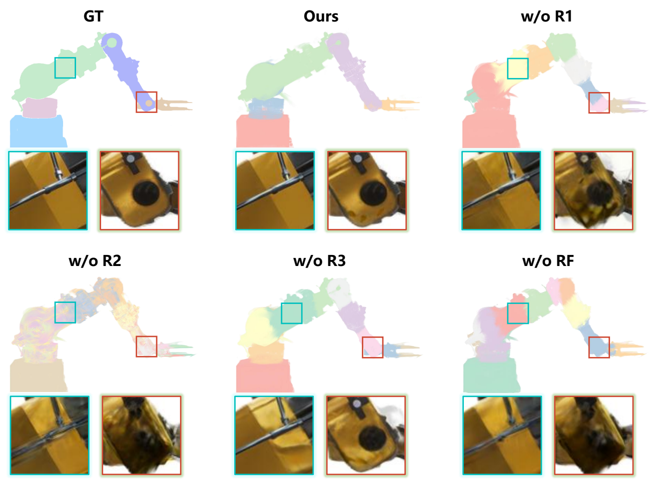


Fig. 7: Ablation studies for regularization and refinement effects.

belonging to the same part, resulting in incorrect segmentation and motion reconstruction. Moreover, removing hierarchical orthogonality (w/o R3) produces non-orthogonal motion decomposition and excessive over-clustering.

Refinement. We examine the effects of the proposed motion hierarchal Gaussian refinement. We use the initialized hierarchical motion tree to reconstruct the local motion when disabling the motion hierarchical Gaussian refinement (w/o RF). For a fire comparison, we train the local motion for 110k in total. As shown in Fig.7, disabling the refinement results in incorrect Gaussian segmentation as well as poor reconstruction quality due to the deformation field being less effective in motion representation.

Table 5: Ablation study of reconstruction and segmentation quality.

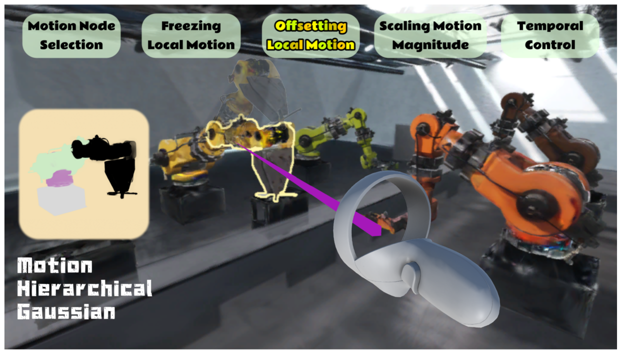
method	Ours	w/o R1	w/o R2	w/o R3	w/o RF
PSNR↑	**38.09**	29.47	25.07	31.08	25.95
LPIPS↓	**0.0053**	0.0213	0.0359	0.0166	0.0331
mIoU↑	**0.7714**	0.4297	0.4547	0.4929	0.5093

5 User Study

We conducted a user study to evaluate the efficiency and accuracy of motion control on the *Robotic* scenes.

5.1 User Study Design

Participants and Setup. 16 participants (12 males and 4 females, aged between 21-30) were recruited in this study, and all of them have had experiences in VR HMDs. Each participant wore an PICO 4 Ultra HMDs for the user study. The research was performed under the oversight of Biology and Medical Ethics Committee of Beihang University,



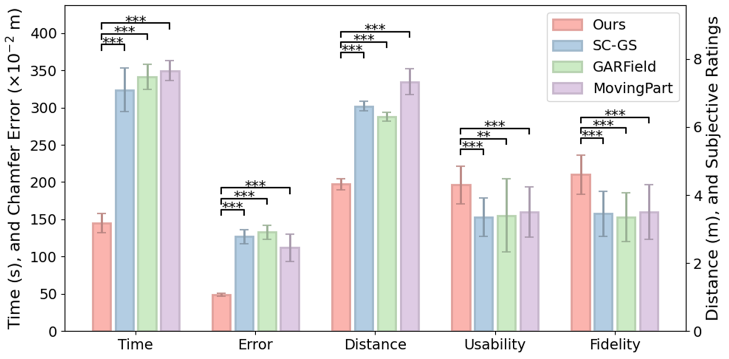


Fig. 8: Task and results of user study. Participants are required to control *Robotic*'s motion (left). Our method achieves significant reduction in completion time, manipulation distance, and motion control error, and significant improvement in manipulation usability and motion fidelity (right).

with protocol number BM20240277. Consent from the human subjects in the research was obtained.

Conditions. The conditions included our method (Ours), SC-GS [11], GARField [67], and MovingPart [13]. For SC-GS, we retain its original motion control mechanism. For GARField and MovingPart, they lack motion control capability. To enable fair comparison in VR, they were minimally extended to support part-level motion control. These modifications only allow completion of the same user study tasks and preserve their original interaction design. Specifically, we first identify the Gaussians to be manipulated based on segmentation, and then apply motion control by adjusting the global motion of the corresponding Gaussian group.

Task and Procedure. As shown in the left of Fig. 8, the user study consists of a single task with five operations. In Operations 1–4, participants first selected a Robotic part and then performed motion control according to given instructions using either our method or the baselines. Operation 5 required participants to freely control local motions of the Robotic arm. The segmentation result is shown in the lower left, with the selected part highlighted in black. Each test includes five repetitions of Operations 1–5, with different *Robotic* motions in each repetition. The order of Operations 1–4 is randomized using a Latin square design, and Operation 5 starts only after all four operations are completed. A 2-minute rest is provided after Operations 1–4, followed by a 10-minute break before Operation 5. After each operation, participants reported manipulation usability and motion fidelity.

Metrics. We report three objective and two subjective metrics. Task completion **time** (seconds) measures the duration from task start to completion. Manipulation **distance** (meters) measures the total movement of the operated handles. Motion control **error** (centimeters) is computed as the average Chamfer Distance over all timestamps. Manipulation **usability** and motion **fidelity** are rated on a 5-point scale from 5 (best) to 1 (worst).

Statistical analysis. We compared different conditions by first assessing data normality using the Shapiro–Wilk test. If the data followed a normal distribution, comparisons were conducted using repeated-measures ANOVA; otherwise, the Wilcoxon signed-rank test was applied. In addition to p-values, effect sizes were estimated using Cohen's d, with qualitative interpretations of Huge ($d > 2.0$), Very Large ($2.0 > d > 1.2$), Large ($1.2 > d > 0.8$), Medium ($0.8 > d > 0.5$), Small ($0.5 > d > 0.2$), and Very Small ($0.2 > d > 0.01$).

5.2 Results and Discussion

As shown in the right of Fig.8, we calculate the average values of different metrics over all methods, and use the p-value and Cohen's d to estimate the difference between the comparison conditions and Ours. The results indicate a significant improvement in efficiency and accuracy of motion control. Our method reduces the time compared to the SC-GS, GARField and MovingPast by 55%, 57%, and 58% for the five operations (all p-values are < 0.001 and the effect sizes are all Huge); reduces the operation error by 58%, 62%,and 53% (all p-values are < 0.001 and the effect sizes are from Very Large to Huge); and reduces the operation distance by 33%, 32%, and 37% (all p-values are < 0.001 and the effect sizes are all Huge). Besides, our method significantly enhanced user immersion and realism compared to SOTA methods. These results demonstrate that compared to previous methods, our method significantly enhances the efficiency of motion control.

Our user study involves a limited number of participants and an imbalanced gender distribution. This study is intended to evaluate the usability and effectiveness of the proposed motion control method and associated VR interaction operations. Despite these limitations, the observed performance improvements are consistent across participants and tasks, with large effect sizes, indicating clear advantages of the proposed approach for VR motion control. A larger and more demographically diverse user study will be explored in future work to further validate these findings.

6 Conclusion

We have proposed a motion hierarchical Gaussian based dynamic control method in VR. A motion hierarchical Gaussian representation was introduced and initialized according to semantic and deformation information. A consistency and orthogonality regularization based motion hierarchical decomposition method was proposed to optimize the local motion. A local motion analysis based refinement method was introduced to optimize the motion hierarchy. A set of motion control operations was designed to effectively interact with the motion hierarchical Gaussian. Extensive experiments on challenging datasets verify that our method achieves high quality motion reconstruction, accurate motion decomposition, real-time, intuitively and immersive motion control in VR.

Our method has several limitations. First, we assume that the clustered Gaussian groups and motion relationship between different groups are time-consistent in our method, which would be invalid when the topology of the object changes, e.g., a cookie is broken into multiple pieces. One future exploration is to incorporate temporally dynamic segmentation properties and develop adaptive cluster strategies to support such cases. Second, our method focuses on motion patterns that can be effectively modeled by structured local motions, whereas fluid motions are continuous, our current hierarchical motion representation is not applicable to such continuous fields. Thus, another future work is to extend our motion formulation and introduce new representations and regularization terms specific to fluid-like motions. The third limitation is that the reconstruction quality for unobserved views can be inherently limited due to the constraints of monocular dynamic input. Even if certain regions are observed at other time, dynamic scene changes prevent their accurate reconstruction at the current time. Future work could address this limitation by incorporating geometric and spatial consistency constraints across time and space, enabling more consistent reconstruction for unobserved views.

Acknowledgments

This work is supported by the National Natural Science Foundation of China through Projects U25A20441, 62507002 and 62372026, the National Key R&D plan 2025YFE0203800, the New Generation Artificial Intelligence-National Science and Technology Major Project 2025ZD0124000, and the Fundamental Research Funds for the Central Universities.

References

[1] 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. 1, 2, 4, 5

[2] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *European Conference on Computer Vision*, pages 162–179. Springer, 2024. 1, 4

[3] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21676–21685, 2024. 1, 2

[4] Seokhun Choi, Hyeonseop Song, Jaechul Kim, Taehyeong Kim, and Hoseok Do. Click-gaussian: Interactive segmentation to any 3d gaussians. In *European Conference on Computer Vision*, pages 289–305. Springer, 2025. 1, 2

[5] Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, et al. Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–1, 2024. 1, 2

[6] Zhaofeng Luo, Zhitong Cui, Shijian Luo, Mengyu Chu, and Minchen Li. Vr-doh: Hands-on 3d modeling in virtual reality. *ACM Trans. Graph.*, 44(4), 2025. 1, 2

[7] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4389–4398, 2024. 1, 2

[8] 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. 1, 2, 4, 5, 7

[9] 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. 1, 2

[10] Deqi Li, Shi-Sheng Huang, Zhiyuan Lu, Xinran Duan, and Hua Huang. St-4dgs: Spatial-temporally consistent 4d gaussian splatting for efficient dynamic scene rendering. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 1

[11] 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. 1, 2, 7, 9

[12] Isaac Labe, Noam Issachar, Itai Lang, and Sagie Benaim. Dgd: Dynamic 3d gaussians distillation. In *European Conference on Computer Vision*, pages 361–378. Springer, 2024. 1, 2, 4, 7

[13] Kaizhi Yang, Xiaoshuai Zhang, Zhiao Huang, Xuejin Chen, Zexiang Xu, and Hao Su. Movingparts: Motion-based 3d part discovery in dynamic radiance field. *arXiv:2303.05703*, 2023. 1, 7, 9

[14] 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. 1, 2

[15] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. *arXiv preprint arXiv:2312.00112*, 2023. 1, 2

[16] Zhen Xu, Yinghao Xu, Zhiyuan Yu, Sida Peng, Jiaming Sun, Hujun Bao, and Xiaowei Zhou. Representing long volumetric video with temporal gaussian hierarchy. *ACM Transactions on Graphics (TOG)*, 43(6):1–18, 2024. 1, 2

[17] Yiming Liang, Tianhan Xu, and Yuta Kikuchi. HiMoR: Monocular deformable gaussian reconstruction with hierarchical motion representation. In *CVPR*, 2025. 1, 2

[18] Guikun Chen and Wenguan Wang. A survey on 3d gaussian splatting. *arXiv preprint arXiv:2401.03890*, 2024. 2

[19] Tong Wu, Yu-Jie Yuan, Ling-Xiao Zhang, Jie Yang, Yan-Pei Cao, Ling-Qi Yan, and Lin Gao. Recent advances in 3d gaussian splatting. *Computational Visual Media*, 10(4):613–642, 2024. 2

[20] 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. *Communications of the ACM*, 65(1):99–106, 2021. 2

[21] 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. 2, 5

[22] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021. 2, 5

[23] Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. Tensor4d: Efficient neural 4d decomposition for high-fidelity dynamic reconstruction and rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16632–16642, 2023. 2

[24] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. pages 5865–5874, 2021. 2

[25] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12959–12970, 2021. 2

[26] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9421–9431, 2021. 2

[27] Xiang Guo, Jiadai Sun, Yuchao Dai, Guanying Chen, Xiaoqing Ye, Xiao Tan, Errui Ding, Yumeng Zhang, and Jingdong Wang. Forward flow for novel view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16022–16033, 2023. 2

[28] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021. 2

[29] Sungheon Park, Minjung Son, Seokhwan Jang, Young Chun Ahn, Ji-Yeon Kim, and Nahyup Kang. Temporal interpolation is all you need for dynamic neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4212–4221, 2023. 2

[30] Jia-Wei Liu, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Devrf: Fast deformable voxel radiance fields for dynamic scenes. *Advances in Neural Information Processing Systems*, 35:36762–36775, 2022. 2

[31] Feng Wang, Sinan Tan, Xinghang Li, Zeyue Tian, Yafei Song, and Huaping Liu. Mixed neural voxels for fast multi-view video synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19706–19716, 2023. 2

[32] Jiwei Shan, Yirui Li, Ting Xie, and Hesheng Wang. Enerf-slam: a dense endoscopic slam with neural implicit representation. *IEEE Transactions on Medical Robotics and Bionics*, 2024. 2

[33] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. New York, NY, USA, 2022. Association for Computing Machinery. 2

[34] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023. 2

[35] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *2024 International Conference on 3D Vision (3DV)*, pages 800–809. IEEE, 2024. 2

[36] Zeyu Yang, Zijie Pan, Xiatian Zhu, Li Zhang, Yu-Gang Jiang, and Philip HS Torr. 4d gaussian splatting: Modeling dynamic scenes with native 4d primitives. *arXiv preprint arXiv:2412.20720*, 2024. 2

[37] Runze Fan, Jian Wu, Xuehuai Shi, Lizhi Zhao, Qixiang Ma, and Lili Wang. Fov-gs: Foveated 3d gaussian splatting for dynamic scenes. *IEEE Transactions on Visualization and Computer Graphics*, 2025. 2

[38] Ashkan Mirzaei, Yash Kant, Jonathan Kelly, and Igor Gilitschenski. Laterf: Label and text driven object radiance fields. In *European Conference on*

Computer Vision, pages 20–36. Springer, 2022. 2

[39] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3835–3844, 2022. 2

[40] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 13779–13788, 2021. 2

[41] Lin Gao, Feng-Lin Liu, Shu-Yu Chen, Kaiwen Jiang, Chunpeng Li, Yukun Lai, and Hongbo Fu. Sketchfacenerf: Sketch-based facial generation and editing in neural radiance fields. *ACM Transactions on Graphics*, 42(4), 2023. 2

[42] Tianhan Xu and Tatsuya Harada. Deforming radiance fields with cages. In *European Conference on Computer Vision*, pages 159–175. Springer, 2022. 2

[43] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: geometry editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18353–18364, 2022. 2

[44] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, Leif Kobbelt, and Lin Gao. Interactive nerf geometry editing with shape priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(12):14821–14837, 2023. 2

[45] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021. 2

[46] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5521–5531, 2022. 2

[47] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *European Conference on Computer Vision*, pages 162–179. Springer, 2025. 2

[48] Qiuhong Shen, Xingyi Yang, and Xinchao Wang. Flashsplat: 2d to 3d gaussian splatting segmentation solved optimally. In *European Conference on Computer Vision*, pages 456–472. Springer, 2025. 2

[49] Lin Gao, Jie Yang, Bo-Tao Zhang, Jia-Mu Sun, Yu-Jie Yuan, Hongbo Fu, and Yu-Kun Lai. Real-time large-scale deformation of gaussian splatting. *ACM Transactions on Graphics (TOG)*, 43(6):1–17, 2024. 2

[50] Antoine Guédon and Vincent Lepetit. Gaussian frosting: Editable complex radiance fields with real-time rendering. In *European Conference on Computer Vision*, pages 413–430. Springer, 2025. 2

[51] Kazunori Yamaguchi, T Kunii, Kikuo Fujimura, and Hiroshi Toriya. Octree-related data structures and algorithms. *IEEE Computer Graphics and Applications*, 4(01):53–59, 1984. 2

[52] Timothy L Kay and James T Kajiya. Ray tracing complex scenes. *ACM SIGGRAPH computer graphics*, 20(4):269–278, 1986. 2

[53] Sagi Katz and Ayellet Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM transactions on graphics (TOG)*, 22(3):954–961, 2003. 2

[54] Marco Attene, Bianca Falcidieno, and Michela Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22(3):181–193, 2006. 2

[55] Yuxue Fan, Yan Huang, and Jingliang Peng. Point cloud compression based on hierarchical point clustering. In *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1–7. IEEE, 2013. 2

[56] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001. 2

[57] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. In *2017 International Conference on 3D Vision (3DV)*, pages 412–420. IEEE, 2017. 2

[58] Olaf Kähler, Victor Prisacariu, Julien Valentin, and David Murray. Hierarchical voxel block hashing for efficient integration of depth images. *IEEE Robotics and Automation Letters*, 1(1):192–197, 2015. 2

[59] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5752–5761, 2021. 2

[60] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 2

[61] Zhide Zhong, Jiakai Cao, Songen Gu, Sirui Xie, Liyi Luo, Hao Zhao, Guyue Zhou, Haoang Li, and Zike Yan. Structured-nerf: Hierarchical scene graph with neural representation. In *European Conference on Computer Vision*, pages 184–201. Springer, 2024. 2

[62] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics (TOG)*, 43(4):1–15, 2024. 2

[63] Shengji Tang, Weicai Ye, Peng Ye, Weihao Lin, Yang Zhou, Tao Chen, and Wanli Ouyang. Hisplat: Hierarchical 3d gaussian splatting for generalizable sparse-view reconstruction. *arXiv preprint arXiv:2410.06245*, 2024. 2

[64] Yuheng Jiang, Zhehao Shen, Yu Hong, Chengcheng Guo, Yize Wu, Yingliang Zhang, Jingyi Yu, and Lan Xu. Robust dual gaussian splatting for immersive human-centric volumetric videos. *ACM Transactions on Graphics (TOG)*, 43(6):1–15, 2024. 2

[65] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023. 3, 7

[66] Ho Kei Cheng, Seoung Wug Oh, Brian Price, Alexander Schwing, and Joon-Young Lee. Tracking anything with decoupled video segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1316–1326, 2023. 3

[67] Chung Min Kim, Mingxuan Wu, Justin Kerr, Ken Goldberg, Matthew Tancik, and Angjoo Kanazawa. Garfield: Group anything with radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21530–21539, 2024. 4, 7, 9

[68] Leland McInnes, John Healy, Steve Astels, et al. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11):205, 2017. 4, 5

[69] Trupti M Kodinariya, Prashant R Makwana, et al. Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95, 2013. 5

[70] Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973. 5