

Appendices

The *Appendix* is organized as follows:

- **Appendix A:** Additional discussions on key design choices.
- **Appendix B:** Further implementation details, including the derivation of gradients, compensation in depth promotion, and training hyperparameter settings.
- **Appendix C:** A discussion of the limitations of our approach.

A More Results and Discussions

Full Results on the Neural 3D Video Dataset. Due to space constraints, Table 2 in the main manuscript reports only PSNR results on the Neural 3D Video dataset. For completeness, Table 1 additionally presents reconstruction quality evaluated with SSIM and LPIPS metrics.

Table 1: Per-scene reconstruction quality on Neural 3D Video dataset.

Scene	PSNR (\uparrow)	SSIM (\uparrow)	LPIPS (\downarrow)	Scene	PSNR (\uparrow)	SSIM (\uparrow)	LPIPS (\downarrow)
Coffee Martini	28.91	0.92	0.061	Cook Spinach	33.17	0.96	0.042
Cut Roasted Beef	33.69	0.96	0.044	Flame Salmon	29.50	0.93	0.056
Flame Steak	33.89	0.97	0.032	Sear Steak	34.10	0.97	0.035

Comparison with Other Motion Modeling Works. To further validate the effectiveness of our work, we conduct comparison experiments with two previous works, Hicom [47] and HiMoR [48] which also employ hierarchical motion modeling. Direct empirical comparison with these two methods is challenging, as HiMoR [48] is tailored for monocular dynamic scene reconstruction and Hicom [47] is developed for streamable dynamic scenes. Both are evaluated on domain-specific benchmarks that differ from ours. To enable a fair comparison, we re-implemented their respective hierarchical motion strategies within our framework. As shown in the table below, our learned adaptive tree consistently outperforms these fixed or heuristic hierarchies across two representative scenes, demonstrating the effectiveness of our approach.

Table 2: Comparison with Hicom [47] and HiMoR [48] on two representative scenes.

Method	PSNR (Cut Roasted Beef)	SSIM	PSNR (Sear Steak)	SSIM
HiMoR-reproduced	32.68	0.96	32.25	0.96
Hicom-reproduced	31.93	0.95	32.11	0.96
TreeSplat (ours)	33.69	0.96	34.10	0.97

Why only hierarchical motion? In our formulation, the tree structure is applied solely to the temporal motion $\mu(t)$ of Gaussians, while the temporal rotation $q(t)$ is modeled independently. This design choice is driven by empirical findings: incorporating the tree structure into the rotation does not lead to noticeable performance improvement, yet increases training time. Moreover, this asymmetry aligns with the nature of scene dynamics, translational motions across hierarchically related objects tend to exhibit strong correlations, whereas rotational changes are typically more localized and less structurally dependent. Hence, we restrict the hierarchical modeling to motion, which captures the most meaningful correlations for dynamic reconstruction.

B More Implementation details

B.1 Derivation of Gradients

The decay-weighted aggregation addresses the instability caused by uniform gradient accumulation across deep motion trees. Without attenuation, ancestor nodes may receive disproportionately large gradients, especially in deeper trees, leading to optimization difficulties. To mitigate this, each Gaussian is equipped with a learnable decay factor $\beta_i \in (0, 1]$, which exponentially downweights the influence of deeper ancestors during both forward and backward passes.

The forward motion of Gaussian i at time t is defined as:

$$\hat{\boldsymbol{\mu}}_i(t) = \sum_{k=0}^{\ell_i} \left(\prod_{p=0}^{k-1} \beta_i \right) \sum_{j=1}^B \mathcal{F}_j^{\text{anc}(i,k)} \mathbf{b}_j^\mu(t), \quad (1)$$

where ℓ_i denotes the depth of Gaussian i in the tree, $\mathcal{F}_j^{\text{anc}(i,k)} \in \mathbb{R}^B$ is the motion coefficient vector at the k -th ancestor node of Gaussian i , and $\mathbf{b}_j^\mu(t) \in \mathbb{R}^3$ denotes the shared time-dependent motion basis functions.

Let the upstream gradient be $\mathbf{g}_i = \partial \mathcal{L} / \partial \hat{\boldsymbol{\mu}}_i \in \mathbb{R}^3$. We now derive the gradients with respect to both the motion field and the decay factor.

Gradient w.r.t. motion coefficients. The partial derivative of the loss with respect to an ancestor node's motion coefficients is:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{F}^{\text{anc}(i,k)}} = \left(\prod_{p=0}^{k-1} \beta_i \right) (\mathbf{g}_i^\top \mathbf{b}_1^\mu(t), \mathbf{g}_i^\top \mathbf{b}_2^\mu(t), \dots, \mathbf{g}_i^\top \mathbf{b}_B^\mu(t)) \in \mathbb{R}^B. \quad (2)$$

Gradient w.r.t. decay factor. Since the decay factor β_i influences the aggregation through exponential products, we compute:

$$\frac{\partial}{\partial \beta_i} \left(\prod_{p=0}^{k-1} \beta_i \right) = k \cdot \beta_i^{k-1}, \quad \text{for } k \geq 1; \quad \frac{\partial}{\partial \beta_i} (1) = 0.$$

Then the gradient of the loss with respect to β_i is:

$$\frac{\partial \mathcal{L}}{\partial \beta_i} = \sum_{k=1}^{\ell_i} k \cdot \beta_i^{k-1} \sum_{j=1}^B \mathcal{F}_j^{\text{anc}(i,k)} \cdot (\mathbf{g}_i^\top \mathbf{b}_j^\mu(t)). \quad (3)$$

These derivations show that the decay-weighted aggregation is fully differentiable. Both the motion coefficients \mathcal{F} and the decay factors β_i are optimized jointly with other parameters during training.

To support efficient training on large-scale dynamic scenes, the gradient computation for both motion coefficients and decay factors is implemented as a custom CUDA kernel. The kernel traverses each Gaussian's ancestral path in parallel and accumulates gradients using atomic operations to handle potential write conflicts on shared ancestor nodes. Memory access is optimized via coalesced reads, and per-thread registers are employed for intermediate accumulations to reduce global memory traffic.

B.2 Compensation in depth promotion

In our framework, *Depth Promotion* operation introduces a new internal motion node by promoting an existing Gaussian's leaf node and attaching two newly created child Gaussians. To ensure motion continuity before and after promotion, we adopt a compensation strategy that preserves the identical motion coefficients for the newly created Gaussians.

Let $\mathbf{c}_i^{\text{pre}} \in \mathbb{R}^B$ denote the aggregated (decayed) motion coefficient for Gaussian i before depth promotion, computed as in Eq. 6. After promotion, the original node becomes a parent, and each newly created Gaussian i' and i'' is assigned a fresh leaf node at level $\ell_{i'} = \ell_{i''} = \ell_i + 1$. To maintain consistency, the decayed motion coefficients $\mathbf{c}_{i'}^{\text{post}}$ and $\mathbf{c}_{i''}^{\text{post}}$ after promotion must match $\mathbf{c}_i^{\text{pre}}$.

To achieve this, we compute the discrepancy between the pre-promotion and post-promotion coefficients,

$$\Delta \mathbf{c}_i = \mathbf{c}_i^{\text{pre}} - \mathbf{c}_i^{\text{post}}, \quad (4)$$

and assign the compensation vector $\Delta \mathbf{c}_i$ as the initial coefficient for both new leaf nodes. This ensures:

$$\mathbf{c}_{i'}^{\text{post}} = \mathbf{c}_{i''}^{\text{post}} = \mathbf{c}_i^{\text{pre}}, \quad (5)$$

and thus the aggregated motion $\hat{\boldsymbol{\mu}}(t)$ remains unchanged during tree growth.

This compensation mechanism allows the tree structure to grow hierarchically without disrupting the temporal motion field. As a result, it preserves training stability and ensures that the motion modeling remains continuous across densification steps.

B.3 Hyper-parameters configuration

Table 3: Detailed hyper-parameters configuration of TreeSplat. All training settings of other hyper parameters, including optimization and densification parameters are keep same as original Gaussian Splatting.

Basis MLP	
input width	32
hidden width	512
# hidden layers	3
activation	SiLU
output width	$B \times 3 + B \times 4$
basis number B	10 for DNeRF, and 16 for N3V
Training	
weight β	5e-3
weight β learning rate scheduler	decay from from 5e-3 to 5e-5 in last 5000 iterations
coefficients c_μ learning rate	1.6e-4
coefficients c_q learning rate	5e-4
<i>Leaf Expansion</i> interval	100
<i>Depth Promotion</i> interval	500

C Limitations

Though TreeSplat achieves state-of-the-art performance with fast rendering and compact storage, the introduction of a hierarchical motion tree and decay-weighted aggregation brings moderate overhead during training due to tree traversal and additional parameter optimization. Moreover, our framework learns motion structures by relying solely on photometric consistency. While this design ensures generality, incorporating external motion priors such as monocular depth, optical flow, or segmentation cues could further improve dynamic scene reconstruction quality, especially in scenes with ambiguous or low-texture regions. We leave this integration of external guidance as promising future work.

D Rendered Videos

We render videos for all scenes in the D-NeRF dataset at a resolution of 800×800 , and for all scenes in the Neural 3D Video dataset at a resolution of 1352×1014 . Please refer to the supplementary video for visual results.