OCCAMSPLAT: LEARNING APPEARANCE CAPACITY FOR DYNAMIC GAUSSIAN SPLATTING

Anonymous authors
Paper under double-blind review

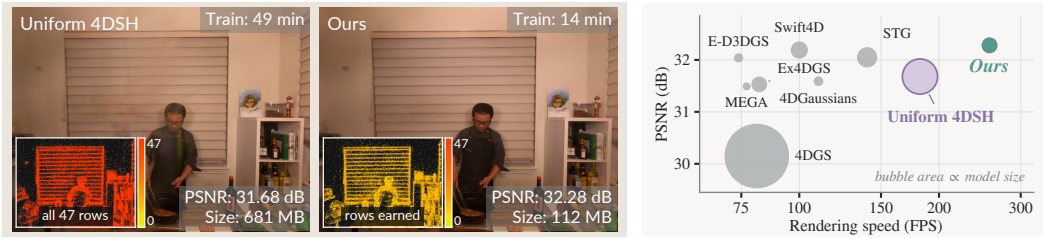

Figure 1: **More appearance capacity is not always better.** Uniform 4DSH gives every Gaussian a full angular–temporal appearance model, whether needed or not. OccamSplat activates proposed capacity only after its joint update passes rendering verification. With just 12.8% of non-DC appearance coefficients active, it improves reconstruction quality over Uniform 4DSH using models about 6× smaller, while rendering at 257 FPS.

ABSTRACT

Capturing view- and time-dependent appearance with explicit Gaussian models can be costly when every primitive receives a full angular–temporal basis. Gaussians contribute unequally to rendered images, and their needs for directional and temporal variation differ. Uniform allocation therefore devotes storage and computation to capacity that may do little for reconstruction. Useful extensions are hard to identify before fitting them, because motion and residual magnitude are indirect cues, and updates to different Gaussians can interact. We introduce OccamSplat, which activates appearance capacity only after verifying that a concrete joint update improves rendering. Starting from base color, we probe candidate four-dimensional spherical-harmonic (4DSH) bands with zero coefficients, preserving the current rendering. Cross-observation gradient agreement yields bounded coefficient updates, accepted only if they jointly lower the rendering loss on two disjoint sets of sampled observations not used to construct them. We evaluate on Neural 3D Video, MeetRoom, and Technicolor light-field. On Neural 3D Video, under the same backbone and densification configuration, our method improves PSNR over Uniform 4DSH by 0.60 dB with 12.8% of non-DC coefficient rows active, while rendering at 257 FPS.

1 INTRODUCTION

Reconstructing dynamic scenes from 2D images lets viewers revisit an event from viewpoints that were never recorded. For immersive AR/VR, these novel views must be photorealistic and respond in real time as both the viewer and the scene move. Dynamic Gaussian Splatting brings this goal within reach through explicit scene primitives and efficient rasterization (Kerbl et al., 2023; Wu et al., 2024; Yang et al., 2024a). Beyond geometry, however, its efficiency also depends on how much appearance complexity each Gaussian carries.

Appearance can be predicted by a shared neural decoder (Li et al., 2024; Zhang et al., 2025a) or represented explicitly on each Gaussian. Spherical harmonics (SH) take the latter approach, expressing color as a constant base plus directional terms (Kerbl et al., 2023). 4D spherical harmonics (4DSH) extend SH with terms that describe color changes over time (Yang et al., 2024a). The temporal terms increase the number of appearance coefficients by a factor of 3. Uniform 4DSH assigns this

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

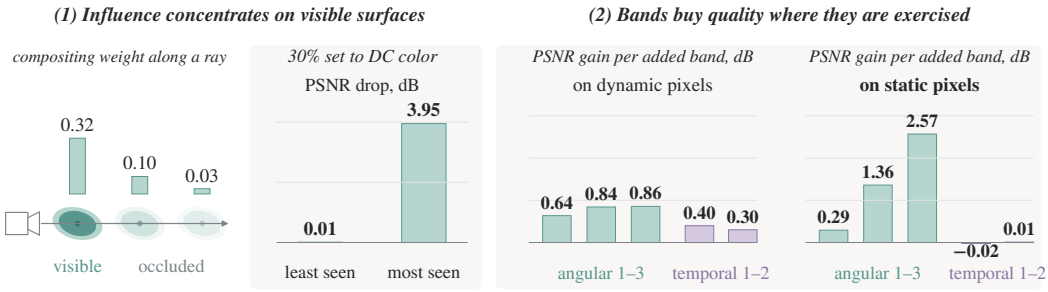

Figure 2: **Appearance needs vary across Gaussians and variation modes.** (1) In a uniform 4DSH model, reducing the least-seen 30% of Gaussians to base color barely affects reconstruction quality, whereas the same restriction on the most-seen 30% causes a large drop. (2) Angular bands improve both temporally stable and changing regions, while temporal bands mainly improve changing ones.

full model to every Gaussian, giving a rarely visible primitive the same capacity as one responsible for a changing surface highlight.

Across datasets, two observations challenge this uniform allocation (Fig. 2): 1) Gaussians behind visible surfaces often have little influence on the rendered image. Reducing the least-seen 30% of Gaussians to base color leaves reconstruction quality nearly unchanged, suggesting that much of their appearance capacity is unnecessary. 2) Directional color variation is needed more widely than temporal color variation. In our scenes, directional terms help in both static and dynamic image regions, whereas temporal terms help mainly in dynamic ones. Appearance requirements therefore vary across both Gaussians and types of variation, yet uniform 4DSH allocates every term to every Gaussian.

A naive strategy is to give less appearance capacity to Gaussians behind visible surfaces and more to those on the surfaces. However, a Gaussian hidden from one viewpoint may be exposed from another, so its contribution must be assessed across observations. Motion is also an incomplete guide to temporal appearance. A moving surface need not change color over time, whereas the lighting on a stationary surface may change. These cues therefore motivate selective capacity without specifying how to allocate it. Allocation should instead depend on whether additional appearance capacity improves reconstruction across views and time.

The key challenge is to find promising updates to candidate bands and verify their joint rendering gain before optimization. Prior work raises SH order using weighted rendering residuals (Liu et al., 2025), but residual magnitude alone does not predict the effect of proposed coefficient updates. Gaussians also contribute jointly through alpha compositing, so individually promising coefficient updates may interfere. We therefore use **verified joint rendering gain as the criterion for activating new appearance capacity**.

From a DC-only initialization, OccamSplat probes each Gaussian's next candidate band with its coefficients set to 0. These zero-point probes leave the current rendering unchanged while measuring the candidate band's first-order response to the residual, including the Gaussian's compositing contribution. Cross-observation gradient agreement then selects candidates and scales their bounded coefficient updates by how consistent the update directions are across views and time.

Without further fitting, we render the proposed updates jointly and test them on two disjoint sets of training observations not used to construct the proposal. These finite-update checks account for interactions among candidates that individual probes cannot capture. Only proposals that reduce the loss on both sets activate their bands and enter subsequent optimization. Capacity thus grows according to verified joint rendering gain, without predefined surface labels, motion labels, or a fixed coefficient budget.

Our contributions are:

- We characterize two sources of appearance redundancy in dynamic Gaussian scenes: unequal contributions of primitives to rendered pixels and unequal demand for directional and temporal variation. Motion alone is therefore an insufficient guide to appearance capacity.

- We introduce an evidence-gated model-selection principle for dynamic Gaussian appearance. Additional capacity is assessed before fitting and becomes part of the model only after its joint rendering gain is verified on observations not used to propose it.
- We evaluate reconstruction quality and efficiency on Neural 3D Video, MeetRoom, and Technicolor. On N3DV, our method renders at 257 FPS on average, using 12.8% of non-DC appearance coefficients.

2 Related Work

Dynamic Gaussian appearance. Dynamic Gaussian methods encode time in deformation fields or primitive attributes. Deformable-3DGS deforms canonical geometry with a coordinate MLP (Yang et al., 2024b), whereas 4DGaussians and E-D3DGS also predict time-conditioned changes to appearance attributes (Wu et al., 2024; Bae et al., 2025). Fully explicit methods instead represent dynamics through time-varying primitive parameters (Yang et al., 2024a; Lee et al., 2024b). Others factor view–time appearance through per-Gaussian features and shared decoders (Li et al., 2024; Zhang et al., 2025a), or allocate a temporal branch with a static–dynamic partition (Wu et al., 2025b). To cut 4DSH cost, MEGA pairs a per-Gaussian base color with a shared view–time predictor. We keep the explicit angular–temporal basis and use joint rendering verification to decide which additional bands become active.

Compact and adaptive Gaussian appearance. Appearance redundancy has been addressed both by changing the representation and by varying SH order. Compact-3DGS replaces per-Gaussian SH with a grid-based neural field and separately learns a mask for pruning primitives (Lee et al., 2024a); alternative bases use spherical-Gaussian lobes or learnable Beta kernels (Chen et al., 2026; Liu et al., 2026). Among methods that retain SH, one fits all bands and then keeps a shorter per-Gaussian prefix, chosen using color variation and transmittance-weighted color differences (Papantonakis et al., 2024). EfficientGS instead starts from DC and selectively increases angular order using weighted residuals across training views (Liu et al., 2025). Rate–distortion optimization and Trick-GS learn per-Gaussian band masks (Wang et al., 2025a; Armagan et al., 2025), while StructGS uses structural cues for adaptive SH (Huang et al., 2026b). RD4DGS applies learned angular-band masks to both static and dynamic Gaussians, but its Ex4DGS backbone keeps SH coefficients time-invariant (Lee & Baek, 2025). Existing approaches either remove bands (post-fit truncation or learned masks) or grow SH order from residuals. Residual magnitude reflects current reconstruction error but does not directly indicate whether a specific new band can reduce it consistently across observations. We instead learn compact appearance models by activating proposed bands only after verifying that their joint update improves reconstruction.

3 Preliminaries: 4D Gaussian Splatting

Let G_i denote the i-th 4D Gaussian. It is conditioned on timestamp t_s, projected to the image plane, and alpha-composited in depth order (Yang et al., 2024a). We retain this geometry and rasterization model. The explicit four-dimensional spherical-harmonic (4DSH) color of G_i is

$$\mathbf{c}_i(\mathbf{d}_{i,s}, t_s) = \sum_{k=0}^{L_t} \cos\Big(\tfrac{2\pi k\, \Delta t_{i,s}}{T_{\text{seq}}}\Big) \sum_{l=0}^{L} \sum_{m=-l}^{l} \boldsymbol{\beta}_{i,k,l,m} Y_{lm}(\mathbf{d}_{i,s}). \tag{1}$$

Here $\mathbf{d}_{i,s}$ is the viewing direction, $\Delta t_{i,s} = \mu_{t,i} - t_s$ the offset from temporal mean $\mu_{t,i}$, T_{seq} the sequence duration, Y_{lm} the real SH basis, and $\boldsymbol{\beta}_{i,k,l,m} \in \mathbb{R}^3$ an RGB coefficient. The angular basis has $(L+1)^2$ RGB coefficient rows; its temporal extension has $(L+1)^2(L_t+1)$. With L=3 and L_t=2, the full appearance coefficient matrix therefore has 48 rows instead of 16: one view- and time-independent DC row and 47 non-DC coefficient rows that can be activated during training.

4 OccamSplat

4.1 Verification-gated appearance growth

We represent each Gaussian with DC color and an active prefix of the band sequence (Fig. 3)

$$[\, A_1, A_2, A_3, T_1, T_2 \,], \qquad (b_1, \ldots, b_5) = (3, 5, 7, 16, 16). \tag{2}$$

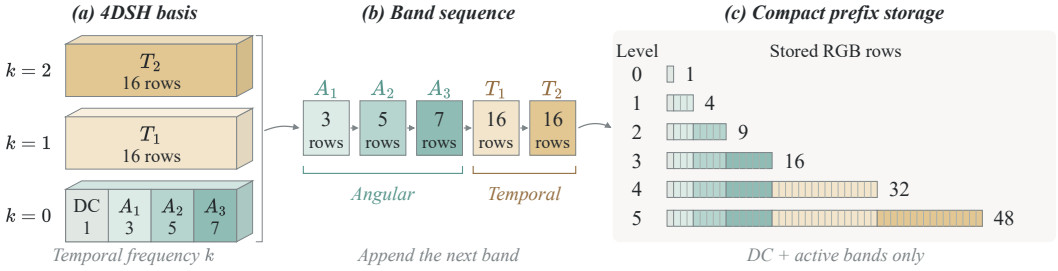

Figure 3: **4DSH bands and per-Gaussian storage.** (a) The 4DGS representation allocates 48 RGB coefficient rows to every Gaussian. (b) The band sequence orders the 47 non-DC rows so that appearance can grow from DC through angular and temporal bands. (c) Compact storage retains only DC and the active prefix and allocates space as bands are activated.

A Gaussian G_i at level $\ell_i \in \{0, \dots, 5\}$ stores the first $\sum_{j \le \ell_i} b_j$ non-DC coefficient rows; later bands are not allocated. A_1–A_3 increase angular order at $k{=}0$; T_1 and T_2 add full angular blocks modulated by the two temporal frequencies. The contiguous prefix lets us store and evaluate only DC and active bands.

Training starts with every Gaussian at level 0 (DC only). Each allocation event resamples three pairwise disjoint sets of training view–time observations: the probe window $\mathcal{W}$, the proposal-check set $\mathcal{R}$, and the audit set $\mathcal{V}$ (Fig. 4). We probe next-band gradients on $\mathcal{W}$ (Sec. 4.2) and use gradient agreement across observations to propose a set of coefficient updates for one selected band (Sec. 4.3). We then check whether this fixed proposal reduces the reconstruction loss on $\mathcal{R}$ and, if so, on $\mathcal{V}$ (Sec. 4.4). If both checks pass, we apply the proposed coefficient updates and activate the next band for each selected Gaussian; otherwise, the model remains unchanged.

4.2 Probing appearance capacity before activation

A hypothetical next-band extension indicates how the current image could be improved. With candidate coefficients set to 0, it reproduces the current rendering, yet its loss gradient can be nonzero. These coefficients are probe variables, not stored parameters of the current model. A single forward and backward pass per observation computes the next-band gradients of all Gaussians at once. We evaluate all candidates on observations from the probe window $\mathcal{W}$, with model parameters θ frozen. For a Gaussian G_i with $\ell_i < 5$, an observation $s \in \mathcal{W}$, and candidate coefficients $\boldsymbol{\beta}_i^+ \in \mathbb{R}^{b_{\ell_i+1} \times 3}$, define

$$\mathbf{g}_i^{(s)} = \nabla_{\boldsymbol{\beta}_i^+} \mathcal{L}_s(\theta; \boldsymbol{\beta}_i^+) \big|_{\boldsymbol{\beta}_i^+ = 0}, \tag{3}$$

where $\mathcal{L}_s(\theta; \boldsymbol{\beta}_i^+)$ is the reconstruction loss of the hypothetical extension and $\mathcal{L}_s(\theta; 0) = \mathcal{L}_s(\theta)$. As in training, the loss is $0.8\,\mathrm{L}_1 + 0.2\,(1 - \mathrm{SSIM})$. Each zero-point probe predicts the first-order loss change from a small update to the candidate coefficients. Unlike a scalar residual magnitude, it retains the response of the specific candidate basis, including its angular and temporal modulation.

Expanding Eq. (3) through alpha compositing reveals how visibility enters the probe. Let $\hat{\mathbf{C}}_p^{(s)} \in \mathbb{R}^3$ be the rendered pixel color and $\boldsymbol{\phi}_i^{+,(s)} \in \mathbb{R}^{b_{\ell_i+1}}$ collect the candidate band's basis values from Eq. (1). The candidate color increment is $(\boldsymbol{\beta}_i^+)^\top \boldsymbol{\phi}_i^{+,(s)}$. With geometry and opacity fixed and color clamping inactive, the chain rule gives

$$\mathbf{g}_i^{(s)} = \sum_p \underbrace{\alpha_{i,p}^{(s)} \prod_{i' \prec i} \left(1 - \alpha_{i',p}^{(s)}\right)}_{w_{i,p}^{(s)}} \boldsymbol{\phi}_i^{+,(s)} \left(\nabla_{\hat{\mathbf{C}}_p^{(s)}} \mathcal{L}_s\right)^\top \Big|_{\boldsymbol{\beta}_i^+ = 0}, \tag{4}$$

where $\alpha_{i,p}^{(s)}$ is the projected opacity of G_i and $i' \prec i$ means that $G_{i'}$ lies in front of G_i on the ray. Each pixel contributes the outer product of the candidate basis and the full image-loss gradient with respect to that pixel's color, scaled by the compositing weight $w_{i,p}^{(s)}$. A Gaussian thus receives little evidence from views in which it is occluded, while remaining eligible in views where it is visible. The gradient reflects both the candidate band's potential to reduce reconstruction loss and the Gaussian's contribution to rendered pixels.

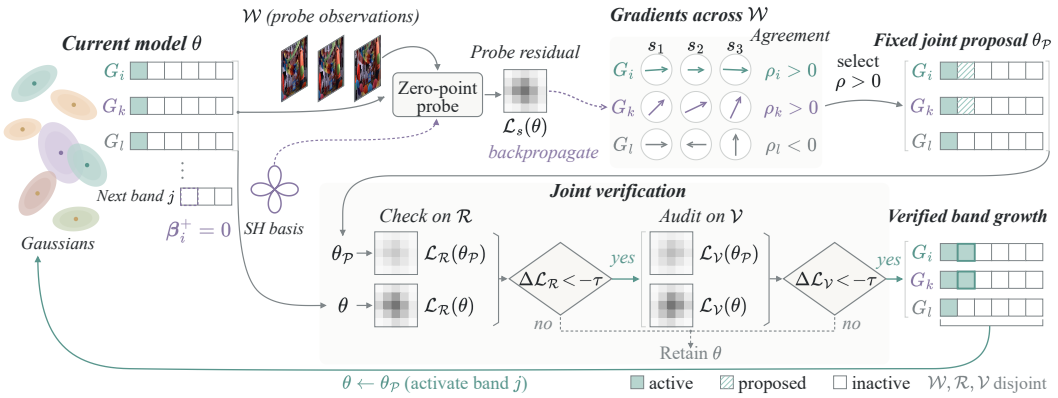

Figure 4: **Overview of our pipeline.** 1) Zero-point probes on $\mathcal{W}$ evaluate next-band gradients without changing the image. 2) Gradient agreement selects candidates and scales their updates. 3) The fixed joint proposal is checked on $\mathcal{R}$ and $\mathcal{V}$; passing both applies the updates and activates bands. The three training sets are pairwise disjoint within each event.

4.3 Proposals from gradient agreement

Given the zero-point probes of Section 4.2, we use gradient agreement across the probe window $\mathcal{W}$ to decide both which candidates to include and how far to update their coefficients. For a Gaussian G_i visible in $m_i \geq 2$ probe observations with nonzero total gradient energy, we measure this agreement by

$$\rho_i = \frac{2\sum_{s<s'} \langle \mathbf{g}_i^{(s)}, \mathbf{g}_i^{(s')} \rangle_F}{(m_i - 1)\sum_{s\in\mathcal{W}} \|\mathbf{g}_i^{(s)}\|_F^2} \in [-1, 1], \tag{5}$$

where $s, s' \in \mathcal{W}$. A positive pairwise inner product means that, to first order, a step along one observation's negative gradient also reduces the other's loss. The score therefore measures the compatibility of coefficient updates suggested by different observations. At each event, we test one band j. We select Gaussians G_i at level $j-1$ with positive gradient agreement, forming the index set $\mathcal{P} = \{i \mid \ell_i = j-1,\ \rho_i > 0\}$.

For each $i \in \mathcal{P}$, let $\bar{\mathbf{g}}_i = |\mathcal{W}|^{-1}\sum_{s\in\mathcal{W}} \mathbf{g}_i^{(s)}$. The update follows the negative mean gradient, with a norm that grows with ρ_i up to a maximum of r:

$$\Delta\boldsymbol{\beta}_i^+ = -r\sqrt{\mathrm{clamp}(\rho_i, 0, 1)}\,\frac{\bar{\mathbf{g}}_i}{\|\bar{\mathbf{g}}_i\|_F}. \tag{6}$$

Since the probes are evaluated with coefficients set to 0, the proposed next-band coefficients equal $\Delta\boldsymbol{\beta}_i^+$. With $\mathcal{P}$ and these coefficients held fixed, we verify the proposal by joint rendering (Sec. 4.4) before any band is activated. Appendix B.1 derives the agreement score and Appendix B.2 the bounded update. Allocation settings are given in Appendix C.1.

4.4 Joint verification and band activation

When the probe loss is differentiable at the zero extension, a nonempty proposal defines a local descent direction on $\mathcal{W}$. At a finite step size, updates can interact through shared pixels and the image loss, so we test the proposal from Section 4.3 by joint rendering. Let $\theta_{\mathcal{P}}$ denote the candidate model that extends θ with next-band coefficients $\Delta\boldsymbol{\beta}_i^+$ for each $i \in \mathcal{P}$ and leaves existing parameters unchanged.

Let $\mathcal{L}_{\mathcal{D}}(\theta)$ denote the mean reconstruction loss over observations in $\mathcal{D}$. The proposal changes this loss by

$$\Delta\mathcal{L}_{\mathcal{D}} = \mathcal{L}_{\mathcal{D}}(\theta_{\mathcal{P}}) - \mathcal{L}_{\mathcal{D}}(\theta), \qquad \mathcal{D} \in \{\mathcal{R}, \mathcal{V}\}, \tag{7}$$

where negative values indicate improvement.

We first evaluate $\Delta\mathcal{L}_{\mathcal{R}}$ on the proposal-check set $\mathcal{R}$. If it falls below $-\tau_{\mathcal{R}}$, we also check the proposal on the audit set $\mathcal{V}$. The thresholds $\tau_{\mathcal{D}} = \tau_{\mathrm{rel}}\mathcal{L}_{\mathcal{D}}(\theta)$ scale each set's baseline loss by a fixed

relative margin $\tau_{\text{rel}} \geq 0$. We update the model parameters only if the loss decreases sufficiently on both sets:

$$\theta^{\text{new}} = \begin{cases} \theta_{\mathcal{P}}, & \Delta\mathcal{L}_{\mathcal{R}} < -\tau_{\mathcal{R}} \wedge \Delta\mathcal{L}_{\mathcal{V}} < -\tau_{\mathcal{V}}, \\ \theta, & \text{otherwise.} \end{cases} \tag{8}$$

On acceptance, the tested coefficient rows are appended to the stored model, and the next band is activated for every Gaussian G_i with $i \in \mathcal{P}$. We then resume standard optimization of geometry and active appearance.

Each accepted joint proposal reduces the mean loss on the sampled $\mathcal{R}$ and $\mathcal{V}$. Repeated sampling and continued mini-batch optimization aim to reconstruct all training views and times. Algorithm 1 summarizes the training procedure.

5 Experiments

5.1 Experimental setup

Datasets and metrics. Main comparisons use 6 Neural 3D Video (N3DV) scenes (Li et al., 2022b) at $1/2$ resolution, with all 300 frames of `cam00` held out. Transfer tests cover 3 MeetRoom scenes (Li et al., 2022a) and 5 Technicolor light-field scenes (Sabater et al., 2017). We report scene-averaged PSNR, DSSIM, AlexNet LPIPS, FPS, model size, training time, and the active non-DC-row share.

Implementation. Our method and Uniform 4DSH use the same 4DGS backbone (Yang et al., 2024a) and FastGS densification (Ren et al., 2026) for 15k iterations. Uniform 4DSH progressively activates the full basis for all Gaussians; ours allocates bands individually. Allocation starts at iteration 500 and repeats every 80 iterations with proposal radius $r = 0.05$. Each event samples $|\mathcal{W}| = 8$ probe, $|\mathcal{R}| = 2$ proposal-check, and $|\mathcal{V}| = 2$ audit training observations without replacement, making the sets pairwise disjoint. Both checks use $\tau_{\text{rel}} = 10^{-4}$, with $\tau_{\mathcal{D}} = 10^{-4}\mathcal{L}_{\mathcal{D}}(\theta)$. All experiments run on an RTX 4090; results are means over 3 runs.

5.2 Efficiency and reconstruction quality

On N3DV, OccamSplat leads the RTX 4090 comparison at 1352×1014 with a mean 257 FPS (Table 1): $1.41\times$ Uniform 4DSH and 2.34–$3.47\times$ the other six baselines. With the shared configuration, it gains 0.60 dB over Uniform 4DSH, uses 112 versus 681 MB uncompressed, and trains in 14 versus 49 min. Compared with Swift4D, ours has similar mean PSNR (32.28 vs. 32.19 dB) and identical rounded LPIPS (0.043), while training in 14 versus 29 min. With unchanged allocator settings, ours improves PSNR over Uniform 4DSH on all 3 MeetRoom scenes at a low active non-DC-row share (Appendix C.5), and reaches 33.77 dB on Technicolor (Appendix C.6).

Table 1: **Quantitative results on N3DV.** FPS is measured at half resolution (1352×1014). Sizes are uncompressed for ours and Uniform 4DSH; ^{c} denotes compressed storage. * marks results from official code or released checkpoints (Appendix D). †Per 50-frame model. ‡Reported in the original paper. Our PSNR has a standard deviation of 0.14 dB across seeds. Per-scene results are in Table 8.

	Reconstruction quality				Efficiency			
Method	PSNR↑	DSSIM_1 ↓	DSSIM_2 ↓	LPIPS↓	FPS↑	Size	#GS↓	Train↓
Reported in the original papers								
STG (Li et al., 2024)	32.05	0.026	0.014	0.044	140	200	0.22M†	5 h
RD4DGS-L6 (Lee & Baek, 2025)	29.66	0.034	–	0.069	101	11^c	–	∼2 h
MEGA (Zhang et al., 2025a)	31.49	0.029	0.017	0.057	77	25^c	–	–
Evaluated on our RTX 4090								
4DGaussians* (Wu et al., 2024)	31.59	0.033	0.017	0.056	110	39	0.12M	34 m
E-D3DGS* (Bae et al., 2025)	32.04‡	0.024‡	–	–	74	35‡	–	1.9 h‡
Swift4D* (Wu et al., 2025b)	32.19	0.030	0.015	0.043	100	139	0.31M	29 m
4DGS* (Yang et al., 2024a)	30.15	0.031	0.017	0.049	81	2208	3.51M	5.1 h
Ex4DGS* (Lee et al., 2024b)	31.53	0.034	0.018	0.060	82	116	0.27M	0.6 h‡
Mango-GS* (Huang et al., 2026a)	28.55	0.051	0.030	0.077	79	108	0.37M	–
Uniform 4DSH	31.68	0.033	0.018	0.047	182	681	1.11M	49 m
OccamSplat	32.28	0.029	0.015	0.043	257	112	0.79M	14 m

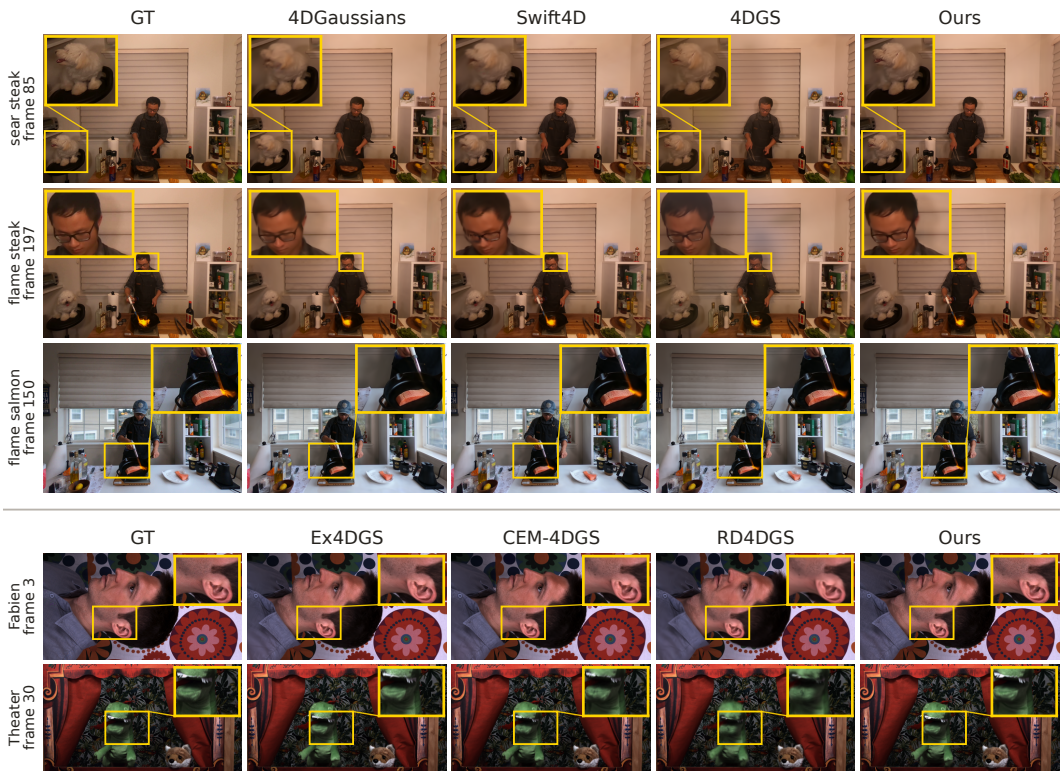

Figure 5: **Qualitative comparison on N3DV (top) and Technicolor (bottom).** Insets show finer textures and sharper boundaries with OccamSplat. All scenes: Appendix D.1.

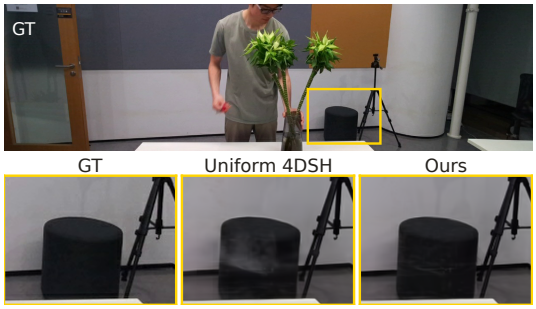

Figure 6: **MeetRoom "trimming", frame 50.**

OccamSplat recovers fine textures more faithfully (Figure 5). For example, it preserves the dog's fur detail in "sear steak", where 4DGaussians and Swift4D blur the head region and 4DGS smears the fur. It also preserves sharper boundaries, as in "Theater", where the puppet's teeth remain distinct while Ex4DGS, CEM-4DGS, and RD4DGS-L6 blur or smear them. Uniform 4DSH instead adds structure that is not in the scene: in the MeetRoom scene "trimming" (Figure 6), it leaves a whitish haze across the dark stool, whereas OccamSplat keeps the stool close to its ground-truth tone. The full basis lets Uniform 4DSH fit view- and time-specific residuals, while OccamSplat activates a band only after its joint update passes rendering verification.

5.3 Allocator ablations

We ablate verification, agreement-based step scaling, and positive-agreement filtering to measure their effects on reconstruction quality, active capacity, and training time (Table 2).

Ablating verification. Removing the audit on $\mathcal{V}$ (Full to A1) lowers PSNR by 0.35 dB and raises the active non-DC-row share from 12.8% to 25.0%. Removing the remaining check on $\mathcal{R}$ (A1 to A2) costs another 0.18 dB and increases this share to 58.6%. Under the same 15k-iteration budget, A1 and A2 take $1.47\times$ and $3.71\times$ the training time of Full, respectively. Gating capacity growth on joint rendering gains across disjoint training observations yields higher final quality with a smaller active-row share and shorter total training time.

Ablating step scaling and agreement filtering. Replacing the agreement-scaled step $r\sqrt{\rho_i}$ with a fixed radius r (Full to A3) lowers PSNR by 1.02 dB and the active-row share to 7.3%. Because verification tests the proposed coefficients, step scaling also affects band activation. Further removing

Table 2: **Component ablations on N3DV.** ✓/✗: on/off. $\mathcal{R}/\mathcal{V}$: proposal check/audit; $\rho > 0$: agreement filter; $\sqrt{\rho}$: step scaling ($r\sqrt{\rho_i}$ vs. r). Rows: active non-DC share; Train: relative to A0.

Variant	$\mathcal{R}$	$\mathcal{V}$	$\rho>0$	$\sqrt{\rho}$	PSNR↑	Rows (%)↓	Train (×)↓
A0 (Full)	✓	✓	✓	✓	32.28	12.8	1.00
A1 ($\mathcal{R}$ only)	✓	✗	✓	✓	31.93	25.0	1.47
A2 (Direct activation)	✗	✗	✓	✓	31.75	58.6	3.71
A3 (Fixed radius)	✓	✓	✓	✗	31.26	7.3	0.91
A4 (No ρ filter)	✓	✓	✗	✗	29.30	11.4	0.95

the positive-agreement filter (A3 to A4) costs another 1.96 dB and raises the row share to 11.4%, worsening both quality and allocation efficiency.

Adaptive-SH allocation. We compare two allocator adaptations on a shared 4DGS backbone with the same five appearance bands, matching elapsed time and final active non-DC-row counts. EfficientGS-style growth selects band additions by compositing-weighted residual ranking (Liu et al., 2025), while learned band masks optimize independent activations with a row-weighted sparsity penalty (Wang et al., 2025a; Lee & Baek, 2025). OccamSplat achieves PSNR gains of 0.42 dB over residual-ranked growth and 0.35 dB over learned masks, with lower LPIPS than both (Table 3). Appendix C.3 details the adaptations and budget-matching procedure.

Table 3: **Adaptive-SH comparison on N3DV.** $B/B^\star$: active non-DC-row count relative to the reference; means over scenes and three seeds.

Allocator	$B/B^\star$	PSNR↑	LPIPS↓
EfficientGS-style growth	1.006	31.78	0.049
Learned band masks	0.997	31.85	0.046
OccamSplat	1.000	32.20	0.044

5.4 Allocation quality and learned structure



Figure 7: **Learned capacity placement improves reconstruction quality.** Bars show N3DV means after the same appearance refit with fixed geometry. Random, visibility-ranked, and learned maps share a capacity-level distribution. Percentages denote active non-DC-row shares; dashed lines mark learned-allocation values.

Where appearance capacity is allocated. We compare the learned map with random and visibility-ranked reassignments that preserve its capacity-level distribution and total non-DC-row count. All three refit appearance for 3k iterations from shared DC and zero-initialized non-DC coefficients, with geometry, motion, opacity, and topology fixed. Random reassignment lowers quality, and visibility ranking recovers only part of the loss (Fig. 7). The learned map also outperforms uniform A_2 with fewer coefficients, demonstrating gains beyond sparsity and visibility ranking.

Learned allocation patterns. Mean active non-DC-row share generally rises with cumulative alpha-compositing weight over training observations (Fig. 8). These weights scale pixelwise probe contributions (Eq. (4)), while agreement and joint verification govern activation. After training, we group Gaussians by center displacement over their temporal support. A displacement threshold yields similar mean shares for low-motion and motion-associated Gaussians (12.8% and 12.9%). The top displacement decile has a higher mean share (17.7%) than the remaining 90% (12.2%; Appendix Table 5(a)).

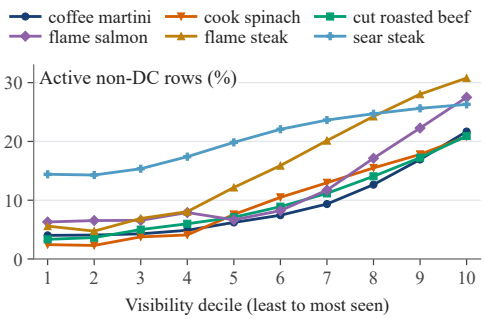

Figure 8: **Capacity and visibility.** Mean active non-DC row share generally rises with visibility.

Reconstruction contribution. In a trained uniform 4DSH model, we measure the reconstruction contribution of non-DC coefficients by restricting the least-seen, a random, or the most-seen 30% of Gaussians to DC. The intervention preserves every Gaussian, its DC color, and all other parameters, removing only non-DC coefficients without refitting. The respective PSNR losses are 0.01, 1.14, and 3.95 dB (Appendix Table 5(b)). Equal removal reduces PSNR most in the most-seen group.

Band granularity and activation order. Our fixed order (Eq. (2)) activates temporal bands T_1, T_2 only after all angular bands A_1–A_3. We test two variants that split each temporal band into four angular blocks ($l = 0, \ldots, 3$). $A \to T$ *(full bands)* follows the original sequence. $A \to T$ *(split T)* allows A_1–A_3 in any order, but tests temporal blocks only after all three are active. *Free A/T (split T)* can test any inactive band or block, including temporal blocks before full angular expansion. Both alternatives increase training time and reduce FPS without a clear PSNR gain (Table 4). Free activation also raises the temporal-to-angular row ratio without improving mean PSNR. We therefore retain full bands and angular-first order, adapting each Gaussian's capacity through its active-prefix length.

Table 4: **Band-order ablation.** T/S: scene-mean ratio of total active RGB rows with $k > 0$ to those with $k = 0, l > 0$; DC excluded. Train: time relative to full bands.

Variant	PSNR↑	T/S	Train↓	FPS↑
$A \to T$ (full bands)	32.28	0.25	1.00	257
$A \to T$ (split T)	32.30	0.21	1.06	249
Free A/T (split T)	32.14	0.79	1.18	218

5.5 Failure cases and limitations

In D-NeRF "lego" (Pumarola et al., 2021), our 4DGS variant produces floating artifacts around moving regions (Figure 9). A monocular sequence observes each instant from only one viewpoint, leaving depth and motion weakly constrained (Liang et al., 2025). Our allocator forms and verifies appearance proposals with geometry and motion fixed, so both stages depend on the current spatial estimates. An appearance update can therefore compensate for errors in these estimates and reduce rendering loss on both verification sets while leaving misplaced Gaussians uncorrected. Our formulation adds no explicit geometric or motion-consistency constraints to resolve this ambiguity. We tried integrating our method with SC-GS (Huang et al., 2024) and found marginal acceleration. Our allocator reduces coefficient count and color evaluation cost by activating fewer bands, but SC-GS's per-frame deformation limits overall speedup.



Figure 9: **Failure on D-NeRF "lego".** Floating artifacts in our 4DGS variant.

6 Conclusion

OccamSplat learns angular–temporal appearance capacity by verifying proposed additions before activation. Zero-point probes and cross-observation gradient agreement form bounded updates, tested jointly on two disjoint sets of training observations excluded from proposal construction. On N3DV, our method improves reconstruction over Uniform 4DSH with lower storage and training time under the same backbone and densification configuration, while achieving the highest mean FPS in our same-GPU comparison. At matched training time and active-row count, it also improves reconstruction over residual-ranked growth and learned band masks. Fixed-geometry controls show that learned placement improves quality at matched capacity, and ablations show that verification limits unnecessary growth.

AI USE STATEMENT

We used generative AI tools to edit the manuscript for readability and to help implement part of the code. All AI-assisted text and code were reviewed and verified by the authors. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The training algorithm and derivations are in Appendix B; implementation settings and extended experiments are in Appendix C. All datasets are public. We provide the complete anonymized code in the supplementary material, including the rendering kernels, allocator, experiment configurations, and training and evaluation scripts.

REFERENCES

Anil Armagan, Albert Saà-Garriga, Bruno Manganelli, Mateusz Nowak, and Mehmet Kerim Yucel. Trick-GS: A balanced bag of tricks for efficient Gaussian splatting. In *ICASSP*, pp. 1–5, 2025.

Jeongmin Bae, Seoha Kim, Youngsik Yun, Hahyun Lee, Gun Bang, and Youngjung Uh. Per-Gaussian embedding-based deformation for deformable 3D Gaussian splatting. In *ECCV 2024*, pp. 321–335, 2025.

Satrajit Chatterjee. Coherent gradients: An approach to understanding generalization in gradient descent-based optimization. In *ICLR*, 2020.

Jiacong Chen, Qingyu Mao, Youneng Bao, Xiandong Meng, Fanyang Meng, Ronggang Wang, and Yongsheng Liang. Motion matters: Compact Gaussian streaming for free-viewpoint video reconstruction. In *NeurIPS*, 2025.

Jiarui Chen, Yikeng Chen, Yingshuang Zou, Ye Huang, Peng Wang, Yuan Liu, Yujing Sun, and Wenping Wang. MEGS^2: Memory-efficient Gaussian splatting via spherical Gaussians and unified pruning. In *ICLR*, 2026.

Woong Oh Cho, In Cho, Seoha Kim, Jeongmin Bae, Youngjung Uh, and Seon Joo Kim. 4D scaffold Gaussian splatting with dynamic-aware anchor growing for efficient and high-fidelity dynamic scene reconstruction. In *AAAI*, pp. 3363–3371, 2026.

Utku Evci, Bart van Merriënboer, Thomas Unterthiner, Max Vladymyrov, and Fabian Pedregosa. GradMax: Growing neural networks using gradient information. In *ICLR*, 2022.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. LightGaussian: Unbounded 3D Gaussian compression with 15x reduction and 200+ FPS. In *NeurIPS*, pp. 140138–140158, 2024.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, pp. 5501–5510, 2022.

Sharath Girish, Tianye Li, Amrita Mazumdar, Abhinav Shrivastava, David Luebke, and Shalini De Mello. QUEEN: QUantized efficient ENcoding of dynamic Gaussians for streaming free-viewpoint videos. In *NeurIPS*, pp. 43435–43467, 2024.

Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. EAGLES: Efficient accelerated 3D Gaussians with lightweight EncodingS. In *ECCV 2024*, pp. 54–71, 2025.

Alex Hanson, Allen Tu, Vasu Singla, Mayuka Jayawardhana, Matthias Zwicker, and Tom Goldstein. PUP 3D-GS: Principled uncertainty pruning for 3D Gaussian splatting. In *CVPR*, pp. 5949–5958, 2025.

Cheng-Yuan Ho, He-Bi Yang, Jui-Chiu Chiang, Yu-Lun Liu, and Wen-Hsiao Peng. TED-4DGS: Temporally activated and embedding-based deformation for 4DGS compression. In *WACV*, pp. 6371–6380, 2026.

Tingxuan Huang, Haowei Zhu, Jun-Hai Yong, Hao Pan, and Bin Wang. Mango-GS: Enhancing spatio-temporal consistency in dynamic scenes reconstruction using multi-frame node-guided 4D Gaussian splatting. In *ICLR*, 2026a.

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 *CVPR*, pp. 4220–4230, 2024.

Zexu Huang, Min Xu, and Stuart Perry. StructGS: Adaptive spherical harmonics and rendering enhancements for superior 3D Gaussian splatting. *IEEE TMM*, 28:1499–1510, 2026b.

Moonsoo Jeong, Dongbeen Kim, Minseong Kim, and Sungkil Lee. DC4GS: Directional consistency-driven adaptive density control for 3D Gaussian splatting. In *NeurIPS*, pp. 28144–28174, 2025.

Taeho Kang, Jaeyeon Park, Kyungjin Lee, and Youngki Lee. Clustered error correction with grouped 4D Gaussian splatting. In *SIGGRAPH Asia*, pp. 1–12, 2025.

Zhihui Ke, Yuyang Liu, Xiaobo Zhou, and Tie Qiu. StreamSTGS: Streaming spatial and temporal Gaussian grids for real-time free-viewpoint video. In *AAAI*, pp. 5629–5638, 2026a.

Zhihui Ke, Yuyang Liu, Xiaobo Zhou, and Tie Qiu. DLGStream: Dynamic language-embedded Gaussian splatting for open-vocabulary enabled free-viewpoint video streaming. In *ECCV*, pp. 215–237, 2026b.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4):139:1–139:14, 2023.

Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3D Gaussian splatting as Markov Chain Monte Carlo. In *NeurIPS*, pp. 80965–80986, 2024.

Hyeongmin Lee and Kyungjune Baek. Temporal smoothness-aware rate-distortion optimized 4D Gaussian splatting. In *NeurIPS*, pp. 132174–132204, 2025.

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3D Gaussian representation for radiance field. In *CVPR*, pp. 21719–21728, 2024a.

Junoh Lee, Changyeon Won, Hyunjun Jung, Inhwan Bae, and Hae-Gon Jeon. Fully explicit dynamic Gaussian splatting. In *NeurIPS*, pp. 5384–5409, 2024b.

Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Ping Tan. Streaming radiance fields for 3D video synthesis. In *NeurIPS*, pp. 13485–13498, 2022a.

Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3D video synthesis from multi-view video. In *CVPR*, pp. 5521–5531, 2022b.

Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime Gaussian feature splatting for real-time dynamic view synthesis. In *CVPR*, pp. 8508–8520, 2024.

Yiqing Liang, Mikhail Okunev, Mikaela Angelina Uy, Runfeng Li, Leonidas Guibas, James Tompkin, and Adam W. Harley. Monocular dynamic Gaussian splatting: Fast, brittle, and scene complexity rules. *TMLR*, 2025.

Jinlong Liu, Yunzhi Bai, Guoqing Jiang, Ting Chen, and Huayan Wang. Understanding why neural networks generalize well through GSNR of parameters. In *ICLR*, 2020.

Qiang Liu, Lemeng Wu, and Dilin Wang. Splitting steepest descent for growing neural architectures. In *NeurIPS*, 2019.

Rong Liu, Zhongpai Gao, Benjamin Planche, Meida Chen, Van Nguyen Nguyen, Meng Zheng, Anwesa Choudhuri, Terrence Chen, Yue Wang, Andrew Feng, and Ziyan Wu. Universal Beta Splatting. In *ICLR*, pp. 101275–101294, 2026.

Wenkai Liu, Tao Guan, Bin Zhu, Luoyuan Xu, Zikai Song, Dan Li, Yuesong Wang, and Wei Yang. EfficientGS: Streamlining Gaussian splatting for large-scale high-resolution scene representation. *IEEE MultiMedia*, 32(1):61–71, 2025.

Linjie Lyu, Ayush Tewari, Jianchun Chen, Thomas Leimkühler, and Christian Theobalt. Faster 3D Gaussian splatting convergence via structure-aware densification. In *SIGGRAPH*, pp. 1–10, 2026.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 41(4):102:1–102:15, 2022.

Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3D Gaussian splatting for accelerated novel view synthesis. In *CVPR*, pp. 10349–10358, 2024.

Panagiotis Papantonakis, Georgios Kopanas, Bernhard Kerbl, Alexandre Lanvin, and George Drettakis. Reducing the memory footprint of 3D Gaussian splatting. *PACMCGIT*, 7(1):16:1–16:17, 2024.

Giambattista Parascandolo, Alexander Neitz, Antonio Orvieto, Luigi Gresele, and Bernhard Schölkopf. Learning explanations that are hard to vary. In *ICLR*, 2021.

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *CVPR*, pp. 10318–10327, 2021.

Shiwei Ren, Tianci Wen, Yongchun Fang, and Biao Lu. FastGS: Training 3D Gaussian splatting in 100 seconds. In *CVPR*, pp. 26094–26103, 2026.

Neus Sabater, Guillaume Boisson, Benoit Vandame, Paul Kerbiriou, Frederic Babon, Matthieu Hog, Remy Gendrot, Tristan Langlois, Olivier Bureller, Arno Schubert, and Valerie Allié. Dataset and pipeline for multi-view light-field video. In *CVPRW*, pp. 1743–1753, 2017.

Seungjoo Shin, Jaesik Park, and Sunghyun Cho. Locality-aware Gaussian compression for fast and high-quality rendering. In *ICLR*, 2025.

Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 3DGStream: On-the-fly training of 3D Gaussians for efficient streaming of photo-realistic free-viewpoint videos. In *CVPR*, pp. 20675–20685, 2024.

Henan Wang, Hanxin Zhu, Tianyu He, Runsen Feng, Jiajun Deng, Jiang Bian, and Zhibo Chen. End-to-end rate-distortion optimized 3D Gaussian representation. In *ECCV 2024*, pp. 76–92, 2025a.

Yifan Wang, Peishan Yang, Zhen Xu, Jiaming Sun, Zhanhua Zhang, Yong Chen, Hujun Bao, Sida Peng, and Xiaowei Zhou. FreeTimeGS: Free Gaussian primitives at anytime anywhere for dynamic scene reconstruction. In *CVPR*, pp. 21750–21760, 2025b.

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 *CVPR*, pp. 20310–20320, 2024.

Jiahao Wu, Rui Peng, Jianbo Jiao, Jiayu Yang, Luyang Tang, Kaiqiang Xiong, Jie Liang, Jinbo Yan, Runling Liu, and Ronggang Wang. LocalDyGS: Multi-view global dynamic scene modeling via adaptive local implicit feature decoupling. In *ICCV*, pp. 9519–9529, 2025a.

Jiahao Wu, Rui Peng, Zhiyan Wang, Lu Xiao, Luyang Tang, Jinbo Yan, Kaiqiang Xiong, and Ronggang Wang. Swift4D: Adaptive divide-and-conquer Gaussian splatting for compact and efficient reconstruction of dynamic scene. In *ICLR*, 2025b.

Lemeng Wu, Bo Liu, Peter Stone, and Qiang Liu. Firefly neural architecture descent: a general approach for growing neural networks. In *NeurIPS*, 2020.

Jinbo Yan, Rui Peng, Zhiyan Wang, Luyang Tang, Jiayu Yang, Jie Liang, Jiahao Wu, and Ronggang Wang. Instant Gaussian stream: Fast and generalizable streaming of dynamic scene reconstruction via Gaussian splatting. In *CVPR*, pp. 16520–16531, 2025.

Jiayu Yang, Weijian Su, Songqian Zhang, Yuqi Han, Jinli Suo, and Qiang Zhang. Scale-GS: Efficient scalable Gaussian splatting via redundancy-filtering training on streaming content. arXiv:2508.21444, 2025.

Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4D Gaussian splatting. In *ICLR*, pp. 9142–9159, 2024a.

Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3D Gaussians for high-fidelity monocular dynamic scene reconstruction. In *CVPR*, pp. 20331–20341, 2024b.

Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. AbsGS: Recovering fine details in 3D Gaussian splatting. In *ACM MM*, pp. 1053–1061, 2024.

Xinjie Zhang, Zhening Liu, Yifan Zhang, Xingtong Ge, Dailan He, Tongda Xu, Yan Wang, Zehong Lin, Shuicheng Yan, and Jun Zhang. MEGA: Memory-efficient 4D Gaussian splatting for dynamic scenes. In *ICCV*, pp. 27828–27838, 2025a.

Zheng Zhang, Wenbo Hu, Yixing Lao, Tong He, and Hengshuang Zhao. Pixel-GS: Density control with pixel-aware gradient for 3D Gaussian splatting. In *ECCV 2024*, pp. 326–342, 2025b.

A Additional related work

Several methods use rendering signals to guide densification and pruning. AbsGS reduces cancellation of positional gradients (Ye et al., 2024), Pixel-GS weights view contributions by Gaussian pixel coverage (Zhang et al., 2025b), and DC4GS uses the directional consistency of pixelwise positional gradients to guide splitting and child placement (Jeong et al., 2025). FastGS relies on reconstruction errors across multiple views (Ren et al., 2026). SADGS compares projected Gaussian scales with image frequencies to guide anisotropic splitting (Lyu et al., 2026). PUP 3D-GS prunes by the second-order sensitivity of reconstruction to spatial parameters (Hanson et al., 2025), and 3DGS-MCMC couples stochastic sampling with primitive relocation (Kheradmand et al., 2024). These methods operate on spatial primitives, whereas our allocator controls appearance capacity.

Appearance compression and dynamic representations reduce redundancy in complementary ways. LightGaussian prunes, distills SH to a lower degree, and quantizes (Fan et al., 2024). EAGLES learns quantized latent attributes, including non-DC SH coefficients (Girish et al., 2025), and C3DGS uses sensitivity-aware codebooks with quantization-aware fine-tuning (Niedermayr et al., 2024). LocoGS shares attributes through spatial fields and learns SH-degree masks whose cumulative products form contiguous prefixes (Shin et al., 2025). In dynamic scenes, LocalDyGS decodes temporal Gaussians from static features and a shared dynamic residual field (Wu et al., 2025a). TED-4DGS entropy-codes attributes and learns a per-anchor temporal activation on top of a shared deformation bank (Ho et al., 2026); its canonical color is time-invariant, and the activation controls when anchors contribute. FreeTimeGS gives each primitive its own temporal center, velocity, and temporal opacity while retaining directional SH for color (Wang et al., 2025b).

Gradient-guided capacity growth has also been studied in neural networks. GradMax initializes added neurons to preserve the network output while maximizing new-weight gradients through an SVD (Evci et al., 2022). Splitting steepest descent uses the eigenstructure of a splitting matrix to select neurons and symmetric splitting directions (Liu et al., 2019). Firefly optimizes candidate expansions within a bounded functional neighborhood and selects additions using Taylor-based contribution scores under a growth budget (Wu et al., 2020). Our zero-point probes share this use of derivatives to assess function-preserving extensions. We grow complete angular–temporal bands in each Gaussian's fixed appearance basis, whose gradients incorporate both the candidate basis and its contribution through alpha compositing (Eq. (4)).

Gradient agreement across observations also connects to generalization. AND-mask filters coordinate updates by gradient-sign agreement across environments (Parascandolo et al., 2021). GSNR relates generalization to the ratio of a parameter's squared mean gradient to its gradient variance (Liu et al., 2020). Coherent Gradients decomposes the squared norm of a summed gradient into self terms and cross-example inner products (Chatterjee, 2020). We use the distinct-pair terms of this same decomposition for each inactive candidate band, normalized by gradient energy and visibility count (Eq. (11)), to select bands and scale bounded updates. Joint rendering then tests the resulting finite proposal before activation.

B Algorithm and theoretical analysis

B.1 Gradient agreement across observations

For the extension of one Gaussian in Eq. (3), fix θ, write $n = |\mathcal{W}|$, and abbreviate $\mathcal{L}_{s'}(\theta; B)$ as $\mathcal{L}_{s'}(B)$. If the loss is differentiable at $B = 0$, a step suggested by observation s changes the loss on another observation according to

$$\mathcal{L}_{s'}(-\eta \mathbf{g}_i^{(s)}) - \mathcal{L}_{s'}(0) = -\eta \langle \mathbf{g}_i^{(s')}, \mathbf{g}_i^{(s)} \rangle_F + o(\eta), \qquad \eta \to 0^+. \tag{9}$$

A positive inner product identifies a direction that locally decreases both observations' losses. Averaging over distinct pairs gives the U-statistic

$$U_i = \frac{2}{n(n-1)} \sum_{s<s'} \langle \mathbf{g}_i^{(s)}, \mathbf{g}_i^{(s')} \rangle_F. \tag{10}$$

Define $S_i = \sum_{s \in \mathcal{W}} \mathbf{g}_i^{(s)}$ and $E_i = \sum_{s \in \mathcal{W}} \|\mathbf{g}_i^{(s)}\|_F^2$. Observations in which G_i is not visible contribute zero gradients, so at most m_i terms are nonzero. Expanding the squared norm gives

$n(n-1)U_i = \|S_i\|_F^2 - E_i$, while Cauchy–Schwarz gives $0 \le \|S_i\|_F^2 \le m_i E_i$. For $m_i \ge 2$ and $E_i > 0$, normalizing by $(m_i - 1)E_i$ yields Eq. (5) and the uniform bound:

$$\rho_i = \frac{n(n-1)U_i}{(m_i-1)E_i} = \frac{\|S_i\|_F^2 - E_i}{(m_i-1)E_i} \in [-1, 1]. \tag{11}$$

Thus U_i and ρ_i have the same sign, and rescaling all gradients by a common nonzero factor leaves ρ_i unchanged.

B.2 Bounded proposals as local descent steps

For $i \in \mathcal{P}$, positive agreement implies $\bar{\mathbf{g}}_i \neq 0$ by Eq. (11). With the chosen radius $a_i = r\sqrt{\rho_i} \le r$, Cauchy–Schwarz gives the steepest-descent step on the linearized mean probe loss:

$$\Delta\boldsymbol{\beta}_i^+ = \operatorname*{arg\,min}_{\|B\|_F \le a_i} \langle \bar{\mathbf{g}}_i, B\rangle_F = -a_i \frac{\bar{\mathbf{g}}_i}{\|\bar{\mathbf{g}}_i\|_F}. \tag{12}$$

Let F be the mean reconstruction loss on $\mathcal{W}$ as a function of hypothetical candidate blocks in $\mathcal{P}$, with the current model parameters fixed. Collect the proposed blocks in Δ. For $\mathcal{P} \neq \varnothing$, differentiability at 0 gives

$$F(\epsilon\Delta) - F(0) = -\epsilon r \sum_{i\in\mathcal{P}} \sqrt{\rho_i}\, \|\bar{\mathbf{g}}_i\|_F + o(\epsilon), \qquad \epsilon \to 0^+. \tag{13}$$

The joint proposal therefore defines a local descent direction on the probe window.

B.3 Finite joint updates and disjoint verification

For finite steps, however, the proposed blocks interact through the image loss. Vectorize the blocks as $\boldsymbol{\delta}_i$ and collect them in $\boldsymbol{\delta}$. With the current model parameters fixed and the loss differentiable to order 2 near 0 (away from L_1 residual kinks), the mean probe loss expands as $\boldsymbol{\delta} \to 0$:

$$F(\boldsymbol{\delta}) - F(0) = \sum_{i\in\mathcal{P}} \langle \bar{\mathbf{g}}_i, \Delta\boldsymbol{\beta}_i^+\rangle_F + \frac{1}{2} \sum_{i,j\in\mathcal{P}} \boldsymbol{\delta}_i^\top H_{ij} \boldsymbol{\delta}_j + o(\|\boldsymbol{\delta}\|_2^2), \tag{14}$$

where H_{ij} is the corresponding block of the Hessian at 0. With geometry, opacity, and renderer branches fixed, the image is affine in the hypothetical candidate coefficients. Let $J_i^{(s)}$ be its block Jacobian and $H_{\mathrm{img}}^{(s)}$ the image-loss Hessian. By the chain rule,

$$H_{ij} = \frac{1}{n} \sum_{s\in\mathcal{W}} \big(J_i^{(s)}\big)^\top H_{\mathrm{img}}^{(s)} J_j^{(s)}. \tag{15}$$

Shared pixels and spatial coupling in the structural loss can make $H_{ij} \neq 0$ for $i \neq j$. These interactions are absent from per-Gaussian agreement scores, so the full finite proposal is evaluated by joint rendering.

An accepted joint proposal reduces the loss on the sampled sets $\mathcal{R}$ and $\mathcal{V}$ at the tested state. The two checks guide allocation during stochastic training but do not guarantee that each accepted proposal reduces the loss over all training observations. Under the differentiability assumption of Appendix B.2, each candidate defines a local descent direction on $\mathcal{W}$, but its standalone and conditional marginal gains are not verified. Later optimization and topology changes can alter the tested state. The sets $\mathcal{W}$, $\mathcal{R}$, and $\mathcal{V}$ are disjoint, but their video observations need not be statistically independent.

B.4 Training algorithm

Algorithm 1 follows the band sequence in Eq. (2), with allocation settings in Appendix C.1.

Algorithm 1 OccamSplat training with verified appearance growth

Require: Training observations, initial model θ_0, allocation schedule, radius r, relative threshold τ_{rel}

Initialize DC-only $\theta \leftarrow \theta_0$, levels $\ell_i \leftarrow 0$, and band-test counts $h_j \leftarrow 0$
for each training iteration **do**
  Update θ by backbone training, including scheduled topology changes
  if an allocation event is due **then**
    Sample pairwise-disjoint training sets $\mathcal{W}$, $\mathcal{R}$, and $\mathcal{V}$
    Probe next-band gradients at 0 on $\mathcal{W}$, holding θ fixed (Eq. (3))
    Compute $\bar{\mathbf{g}}_i$, ρ_i (Eq. (5)); set $\rho_i \leftarrow 0$ if $m_i < 2$ or statistics are invalid
    $\mathcal{P}_j \leftarrow \{i : \ell_i = j - 1,\ \rho_i > 0\}$ for $j = 1, \ldots, 5$
    if all $\mathcal{P}_j = \varnothing$ **then continue**
    $j \leftarrow \arg\min_{k:\mathcal{P}_k \neq \varnothing} h_k$; $\mathcal{P} \leftarrow \mathcal{P}_j$; $h_j \leftarrow h_j + 1$
    $\Delta\boldsymbol{\beta}_i^{+} \leftarrow -r\sqrt{\rho_i}\,\bar{\mathbf{g}}_i / \|\bar{\mathbf{g}}_i\|_F$ for $i \in \mathcal{P}$
    $\theta_{\mathcal{P}} \leftarrow \theta$; append band j to $\theta_{\mathcal{P}}$ with coefficients $\Delta\boldsymbol{\beta}_i^{+}$ for $i \in \mathcal{P}$
    $\Delta\mathcal{L}_{\mathcal{R}} \leftarrow \mathcal{L}_{\mathcal{R}}(\theta_{\mathcal{P}}) - \mathcal{L}_{\mathcal{R}}(\theta)$
    if $\Delta\mathcal{L}_{\mathcal{R}} < -\tau_{\mathrm{rel}}\mathcal{L}_{\mathcal{R}}(\theta)$ **then**
      $\Delta\mathcal{L}_{\mathcal{V}} \leftarrow \mathcal{L}_{\mathcal{V}}(\theta_{\mathcal{P}}) - \mathcal{L}_{\mathcal{V}}(\theta)$
      if $\Delta\mathcal{L}_{\mathcal{V}} < -\tau_{\mathrm{rel}}\mathcal{L}_{\mathcal{V}}(\theta)$ **then**
        $\theta \leftarrow \theta_{\mathcal{P}}$; $\ell_i \leftarrow \ell_i + 1$ for $i \in \mathcal{P}$
      end if
    end if
  end if
end for
return θ, $\{\ell_i\}$

C Extended experiments

C.1 Implementation and evaluation details

Training and allocation settings follow Sec. 5.1.

Uniform 4DSH baseline. Uniform 4DSH allocates the full coefficient tensor and follows a 3DGS-style progressive angular SH-degree schedule (Kerbl et al., 2023). The angular degree starts at 0 and increases globally by one every 1k iterations until reaching 3. The temporal degree then increases at the same interval up to 2. All Gaussians share these active degrees.

Topology and band order. Densification uses gradient thresholds, with splits sampled jointly in $(\mathbf{x}, t)$. In OccamSplat, one child inherits the parent's active prefix after each split, while the others start at level 0. Topology changes reset probe statistics but preserve band-test counts. The band sequence in Eq. (2) follows the native 4DSH layout, which stores all angular rows of one temporal frequency before those of the next.

Allocator implementation. For each Gaussian, we zero its probe gradient in observations where its projected radius is nonpositive. The count m_i includes observations with positive projected radius, while the mean gradient is averaged over the full window $\mathcal{W}$, including zeroed gradients. Eligibility requires $m_i \geq 2$, finite $U_i > 0$, finite $\rho_i \geq 0$, and a finite, nonzero mean-gradient norm. Under the normalization in Eq. (11), $U_i > 0$ is equivalent to positive ρ_i, as required by Algorithm 1. We accumulate distinct-pair inner products directly, without subtracting squared norms. Among bands with eligible candidates, we select the one with the fewest previous tests, breaking ties in favor of the larger index.

Verification and threshold. For each observation in a verification set, we render the current model θ and the fixed joint proposal $\theta_{\mathcal{P}}$ once each. Thus, each verification uses two render calls per observation, independent of the number of candidate Gaussians. We evaluate $\mathcal{V}$ only after the proposal passes the check on $\mathcal{R}$. Both checks use $\tau_{\mathrm{rel}} = 10^{-4}$.

Evaluation protocol. N3DV evaluation uses all 300 held-out `cam00` frames at half resolution (1352×1014). We report scene-averaged PSNR, DSSIM, and AlexNet LPIPS, with means across three runs. DSSIM_1 and DSSIM_2 denote $(1-\mathrm{SSIM})/2$, with SSIM computed using scikit-image on RGB images normalized to $[0, 1]$ and `data_range` set to 1 and 2, respectively. FPS is measured at the same resolution on an RTX 4090; model sizes for OccamSplat and Uniform 4DSH refer to uncompressed storage. Dataset-specific preparation and evaluation settings for MeetRoom and Technicolor are given in Appendices C.5 and C.6.

C.2 Component ablation details

The A0–A4 ablations use the matched 15k budget specified in Sec. 5.1 and the component settings in Table 2. Within each repeat, they share initialization, optimizer, learning-rate schedules, densification rules, observation streams, and random seeds. Geometry and topology remain trainable, and results use the final checkpoints. Metrics are averaged over frames, repeated runs, and then scenes.

Proposal validity. A0–A2 share probing, positive-agreement selection, bounded updates, and band selection, but A2 applies every nonempty, numerically valid proposal. A4 removes the agreement filter but keeps the other eligibility conditions: the next inactive band is tested on a Gaussian with at least 2 visible probe observations, finite statistics, and nonzero mean gradients.

C.3 Adaptive-SH implementation and budget matching

The comparison in Table 3 uses three appearance allocators on N3DV, each with three seeds, sharing the 4DGS backbone, initialization, training observations, optimizer, densification configuration, and full 4DSH basis. The EfficientGS-style rule grew bands by compositing-weighted residual ranking (Liu et al., 2025); the learned-mask rule optimized independent band masks with a row-weighted sparsity penalty (Wang et al., 2025a; Lee & Baek, 2025). Both rules were extended to the five groups in Eq. (2). These are appearance-allocator adaptations, with the shared backbone retained, rather than reproductions of the complete source systems.

For each scene and seed, the elapsed-time budget equaled that of a 15k-iteration OccamSplat reference. All timed runs, including a fresh reference, shared an elapsed-time learning-rate schedule and renderer. Its final active non-DC RGB-row count $B^\star$ defined the capacity target. We tuned each baseline's growth fraction or mask penalty using training data to satisfy $|B - B^\star| \leq \max(16, 0.01B^\star)$, without using our agreement scores or verification decisions. Timing included scoring, warm-up, and verification. Actual row counts, Gaussian counts, and matching errors were retained per run; unmatched runs were identified separately.

C.4 Allocation and band diagnostics

We also conducted the visibility-intervention and band-prefix analyses on other datasets. Figure 2 and Table 5(b) present the N3DV results.

Table 5: **Learned allocation and reconstruction contribution.** (a) Active appearance capacity across Gaussian motion groups on the 6 N3DV scenes. Groups are defined after training using Gaussian center displacement. (b) DC-only interventions in uniform 4DSH. Results are averaged over repeated runs and scenes. "Active rows" is the share of the 47 non-DC rows that are active.

(a) Appearance capacity by Gaussian motion

Motion group	Population	Active rows↓
Low-motion (≤ 2 median scales)	40.2%	12.8%
Motion-associated	59.8%	12.9%
Displacement: lower 90%	90.0%	12.2%
Displacement: top 10%	10.0%	17.7%

(b) DC-only intervention in a uniform 4DSH model

Modified subset (by blending weight)	Subset size	PSNR drop (dB)↓	LPIPS increase↓
Least seen	30%	0.01	0.0001
Random, matched size	30%	1.14	0.0060
Most seen	30%	3.95	0.0259

Allocation controls. For the controls in Sec. 5.4, visibility ranking assigns capacity levels in descending order of cumulative blending weight over training observations. We reset the appearance optimizer states and refit both DC and active non-DC coefficients using identical observations and learning rates. Random-reassignment results are averaged over 3 permutations per model, then over repeated runs and scenes. Uniform A_1 and A_2 use the same refit protocol, with non-DC-row shares $3/47$ and $8/47$.

Motion and visibility diagnostics. After training, we group Gaussians by the displacement of their conditional centers (Table 5(a)). A Gaussian is motion-associated when its conditional center moves more than 2 median Gaussian scales over its temporal support ($\mu_t \pm 2\sigma_t$), clipped to the

sequence. The last two rows split Gaussians at the 90th percentile of this displacement. For the DC-only interventions in Table 5(b), we measure visibility in uniform 4DSH models trained on the 6 N3DV scenes as accumulated blending weight $\sum_{s,p} w_{i,p}^{(s)}$ over 600 training view–time observations spanning all training cameras and the full sequence. Quality changes are evaluated over all 300 frames of the held-out camera and averaged over repeated runs and scenes.

For Fig. 8, we instead use the retained models, accumulating blending weights over 24–25 training view–time observations. Gaussians are grouped into visibility deciles, and each curve reports the mean active share of the 47 non-DC coefficient rows in each decile, averaged over repeated runs.

Evaluation of band prefixes. The cumulative-prefix experiment in Fig. 2 zeroes coefficients above each tested prefix in uniform 4DSH models trained on the 6 N3DV scenes; gains are averaged over repeated runs and scenes. Dynamic regions comprise the 10% of pixels whose ground truth varies most over the 300 frames of the fixed test camera; the remaining pixels form the static regions.

C.5 MeetRoom benchmark

Table 6: **MeetRoom benchmark** (Li et al., 2022a): 3-scene PSNR on undistorted images following Sun et al. (2024); [1]raw-footage preparation, indicative only.

Method	PSNR↑	Method	PSNR↑
StreamRF (Li et al., 2022a)	26.72	IGS-l[1] (Yan et al., 2025)	30.13
Plenoxels (Fridovich-Keil et al., 2022)	27.15	ComGS-s (Chen et al., 2025)	31.49
I-NGP (Müller et al., 2022)	28.10	Scale-GS[1] (Yang et al., 2025)	31.58
3DG-S, per frame (Kerbl et al., 2023)	31.31	StreamSTGS[1] (Ke et al., 2026a)	27.41
3DGStream (Sun et al., 2024)	30.79	DLGStream[1] (Ke et al., 2026b)	27.48
QUEEN-s (Girish et al., 2024)	31.14	Ours, uniform 4DSH	31.16
Swift4D (Wu et al., 2025b)	32.05	**Ours, OccamSplat**	31.60

MeetRoom (Li et al., 2022a) contains 3 sequences of 300 frames at 1280×720, captured with 13 Azure Kinect cameras. Following Sun et al. (2024), we undistort all cameras to a common pinhole model using first-frame COLMAP calibrations, hold out one camera, and average metrics over all frames, repeated runs, and scenes. Allocator settings follow Sec. 5.1. In Table 6, QUEEN-s and ComGS-s numbers come from the re-evaluation by Chen et al. (2025); 3DG-S fits each frame independently. Rows marked [1] use the authors' raw-footage preparation without this undistortion; under that preparation, baselines reported in both settings score 2–5 dB lower (3DGStream: 25.9–28.4 dB). On "discussion", "trimming", and "vrheadset", our method reaches 30.89, 32.19, and 31.71 dB, versus 30.82, 31.96, and 30.70 dB for uniform 4DSH. Active non-DC-row shares are 1.6–3.6%, with image-weighted shares of 12–19%. The image-weighted share is the mean active-row fraction, with each Gaussian weighted by its alpha-compositing contribution to the rendered pixels.

C.6 Technicolor light-field benchmark

Technicolor (Sabater et al., 2017) provides 5 scenes of 50 frames at 2048×1088 from a 4×4 rig. We hold out camera 10 and use the supplied undistorted images and per-camera intrinsics and extrinsics.

Table 7: **Technicolor light-field benchmark.** Reported per-scene PSNR (dB↑) and 5-scene means. * marks results we reproduced with official code or released checkpoints; the other baseline rows list the values reported in the original papers.

Method	"Birthday"	"Fabien"	"Painter"	"Theater"	"Train"	Mean
STG (Li et al., 2024)	32.09	35.70	36.44	30.99	32.58	33.56
MEGA (Zhang et al., 2025a)	32.02	34.89	36.73	31.54	32.69	33.57
RD4DGS-L6 (Lee & Baek, 2025)	31.33	32.90	35.05	30.16	31.55	32.20
CEM-4DGS (Kang et al., 2025)	32.83	35.67	37.08	32.28	32.32	34.04
4D Scaffold-GS (Cho et al., 2026)	32.86	35.15	37.05	32.65	32.86	34.11
Ex4DGS* (Lee et al., 2024b)	32.04	35.20	36.75	31.99	31.93	33.58
OccamSplat	32.44	35.61	36.72	31.89	32.20	33.77

D DETAILED RESULTS

4DGS reproduction. We trained 4DGS using the authors' released code because no reusable N3DV checkpoints were available to us. The original paper reports 32.01 dB mean PSNR (Table 1 of Yang et al., 2024a), whereas our reproduction, marked 4DGS* in Table 8, reaches 30.15 dB, a gap of 1.86 dB. The discrepancy is concentrated in "flame steak" and "sear steak": we obtain 28.34 and 28.84 dB, compared with 34.03 and 33.51 dB in the original paper's Table 4. These deficits of 5.69 and 4.67 dB account for approximately 93% of the mean gap; the other four scenes differ from the original results by at most 0.45 dB. The other methods in Table 8 reach 29.99–34.36 dB across the two steak scenes. We report this as an unresolved reproduction gap rather than an intrinsic limit of 4DGS. The scores alone do not establish whether training diverged.

Sources of the N3DV baseline rows. In Table 1, 4DGaussians* and 4DGS* were retrained with the official code, and every value in their rows is measured on our RTX 4090. Swift4D* quality, FPS, size and Gaussian count come from the released checkpoints; its training time comes from our retraining with the official code. E-D3DGS* FPS is measured on our RTX 4090; its quality, size, and training time are quoted from the original paper, as marked in the table. The STG, RD4DGS-L6, and MEGA rows quote the original papers; their FPS and training times were measured on the authors' hardware. STG reports 40–60 min per 50-frame model, so its entry counts six such models.

Ex4DGS and Mango-GS. Ex4DGS* and Mango-GS* are rendered from the authors' released checkpoints, so we do not measure their training times; Table 1 quotes the training time reported for Ex4DGS. The released Mango-GS models are trained and evaluated at 676×507, and the 31.89 dB reported in the original paper is computed at this resolution; rendering the released weights at 676×507 gives 31.74 dB. Tables 1 and 8 evaluate them at 1352×1014 like every other method, where they reach 28.55 dB, and time Mango-GS per frame, with one deformation step and one rasterization for each rendered frame (79 FPS). With the timing of the original paper, which runs one deformation step per four-frame window and divides its cost among the four frames, the same models reach 184 FPS at 1352×1014.

On the 4 scenes other than "flame steak" and "sear steak", the mean PSNR in Table 8 is 31.34 dB for OccamSplat and 30.93 dB for 4DGS*, a gain of 0.41 dB. The per-scene gains range from 0.19 to 0.68 dB, so our advantage is not confined to the scenes with large reproduction discrepancies.

Table 8: **Per-scene Neural 3D Video results.** We evaluate our method and six baselines on the same RTX 4090, using the metrics of Table 1. Our measurements are means over 3 runs. * marks results from official code or released checkpoints; Ex4DGS and Mango-GS use released checkpoints, so their training times are not measured (Appendix D). Bold marks the best value per metric and scene.

Method / metric	"coffee"	"spinach"	"beef"	"salmon"	"flame"	"sear"	Mean
4DGaussians* (Wu et al., 2024)							
PSNR	**29.23**	33.01	31.64	29.18	33.70	32.77	31.59
$DSSIM_1$	0.048	0.029	0.030	0.047	0.024	0.024	0.033
$DSSIM_2$	0.026	0.014	0.015	0.025	0.011	0.011	0.017
LPIPS	0.081	0.048	0.051	0.079	0.039	0.040	0.056
FPS	98	113	121	97	119	115	110
Size (MB)	**39**	**39**	**39**	**40**	**38**	**38**	**39**
Train (min)	52	24	25	52	25	24	34
Swift4D* (Wu et al., 2025b)							
PSNR	29.12	33.02	33.72	**29.68**	33.63	33.94	32.19
$DSSIM_1$	**0.043**	0.026	**0.024**	0.041	**0.022**	**0.021**	0.030
$DSSIM_2$	**0.024**	0.013	**0.012**	0.022	**0.010**	**0.010**	**0.015**
LPIPS	**0.059**	0.042	0.039	**0.055**	0.032	**0.031**	**0.043**
FPS	88	104	100	87	105	114	100
Size (MB)	112	72	184	112	178	177	139
Train (min)	30	27	28	30	29	29	29
4DGS* (Yang et al., 2024a)							
PSNR	28.39	32.84	33.56	28.93	28.34	28.84	30.15
$DSSIM_1$	0.045	**0.025**	**0.024**	0.043	0.026	0.025	0.031
$DSSIM_2$	0.025	**0.012**	**0.012**	0.024	0.014	0.013	0.017
LPIPS	0.071	0.039	**0.036**	0.066	0.041	0.040	0.049
FPS	48	92	102	49	87	107	81
Size (MB)	2702	2220	1927	2242	2439	1719	2208
Train (min)	378	285	294	341	306	243	308
Ex4DGS* (Lee et al., 2024b)							
PSNR	28.38	32.69	33.16	28.39	33.39	33.19	31.53
$DSSIM_1$	0.050	0.029	0.028	0.047	0.025	0.026	0.034
$DSSIM_2$	0.028	0.014	0.014	0.026	0.012	0.013	0.018
LPIPS	0.085	0.053	0.052	0.081	0.044	0.045	0.060
FPS	61	96	93	62	94	83	82
Size (MB)	130	118	123	128	101	94	116
Mango-GS* (Huang et al., 2026a)							
PSNR	26.28	29.33	29.48	26.08	29.99	30.14	28.55
$DSSIM_1$	0.065	0.048	0.048	0.065	0.042	0.041	0.051
$DSSIM_2$	0.039	0.027	0.027	0.039	0.023	0.022	0.030
LPIPS	0.092	0.075	0.075	0.093	0.064	0.062	0.077
FPS	136	53	54	126	52	54	79
Size (MB)	53	137	142	51	132	131	108
Uniform 4DSH							
PSNR	28.37	**33.47**	32.93	28.72	33.13	33.48	31.68
$DSSIM_1$	0.046	0.026	0.032	0.043	0.023	0.027	0.033
$DSSIM_2$	0.026	0.013	0.018	0.024	0.012	0.015	0.018
LPIPS	0.062	**0.038**	0.047	0.059	0.032	0.044	0.047
FPS	**167**	192	193	168	194	179	182
Size (MB)	705	669	686	708	654	665	681
Train (min)	48	49	49	49	49	48	49
OccamSplat							
PSNR	28.72	33.28	**33.75**	29.61	**33.97**	**34.36**	**32.28**
$DSSIM_1$	0.045	**0.025**	0.025	**0.039**	**0.022**	0.022	**0.029**
$DSSIM_2$	0.026	0.013	**0.012**	**0.021**	0.011	0.011	**0.015**
LPIPS	0.063	**0.038**	0.038	0.057	**0.031**	0.032	**0.043**
FPS	155	**262**	**255**	**247**	**349**	**278**	**257**
Size (MB)	139	93	92	155	98	98	112
Train (min)	**14**	**23**	**13**	**13**	**12**	**11**	**14**
Appearance capacity	9.2%	9.8%	9.7%	12.1%	15.7%	20.4%	12.8%

D.1 QUALITATIVE RESULTS ON EVERY SCENE

Figs. 10–12 show one held-out frame per scene in the layout of Fig. 5. Our method and Uniform 4DSH use 15k-iteration checkpoints. Windows highlight moving content where baselines degrade.

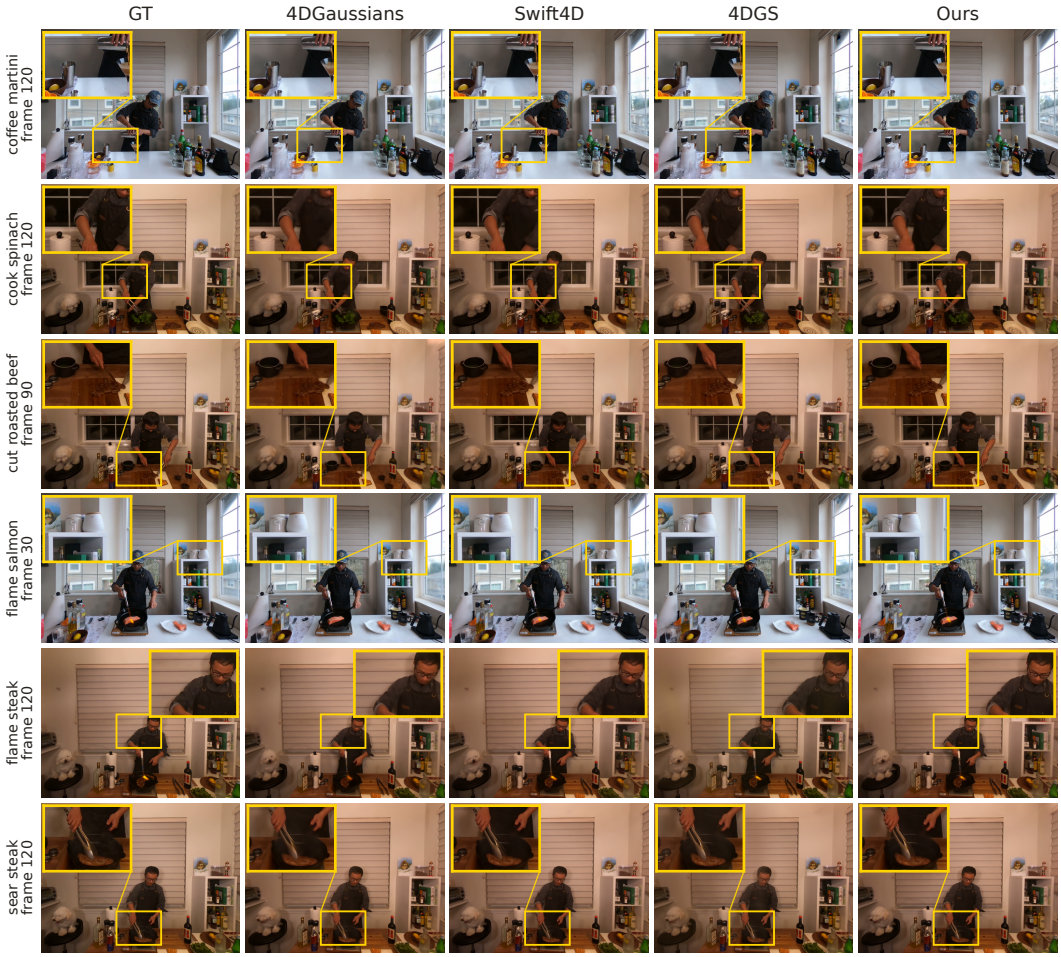

Figure 10: **Neural 3D Video, every scene.** Held-out camera, against 4DGaussians, Swift4D, and 4DGS. Our method leads in every window, by 1.1–2.7 dB over the strongest baseline; the gaps are largest on the face in "flame steak", which 4DGS blurs, and on the tongs and steak in "sear steak".

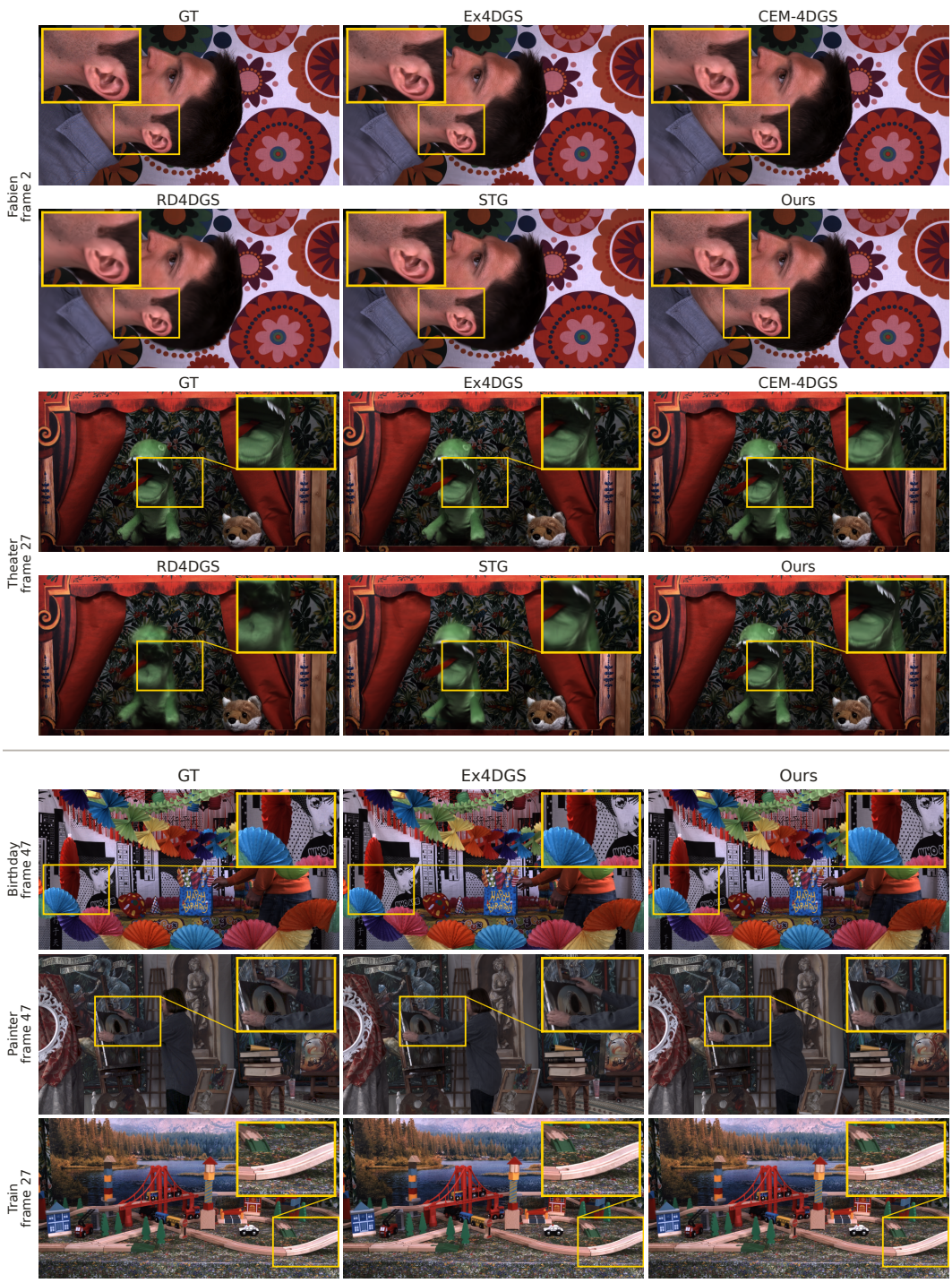

Figure 11: **Technicolor, every scene.** Held-out camera 10. Top: "Fabien" and "Theater", each in two rows of three cells (the ground truth, our reproduction of Ex4DGS, and the released CEM-4DGS model; then our reproductions of RD4DGS-L6 and STG, and OccamSplat). Bottom: the other three scenes against our reproduction of Ex4DGS. For "Fabien" and "Theater" we trained STG and RD4DGS-L6 with the official code and rendered the released CEM-4DGS checkpoints; their held-out PSNRs (STG 35.68/31.10, RD4DGS-L6 32.69/30.08, CEM-4DGS 35.67/32.28 dB) lie within 0.21 dB of the published values listed in Table 7. Per-window gains over the strongest baseline are 1.7–4.8 dB; on "Theater" every baseline smears the moving puppet, and on "Birthday" Ex4DGS blurs the paper fans and the printed dots.

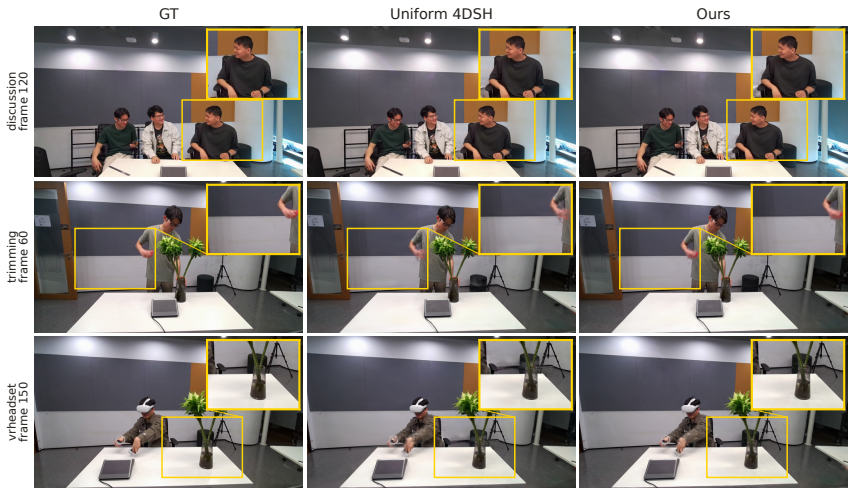

Figure 12: **MeetRoom, every scene.** Held-out camera; Uniform 4DSH is the same backbone with every row active. Per-window gains are 1.7–3.3 dB, largest on the swinging arm in "trimming" and the plant in "vrheadset".

D.2 Temporal consistency

We evaluate 15k-iteration checkpoints of OccamSplat and uniform 4DSH. Selective appearance allocation yields temporal errors close to those of uniform 4DSH on the evaluated diagnostic sequences. For the static held-out cameras, we compute the channel-averaged magnitude of the frame-to-frame change in the rendered and ground-truth videos; the error is their absolute difference, averaged over pixels and time, then across runs. In every evaluated scene, the errors differ by less than 0.03 levels on an 8-bit scale, and the space–time slices in Fig. 13 show coherent trajectories that follow the ground-truth motion.

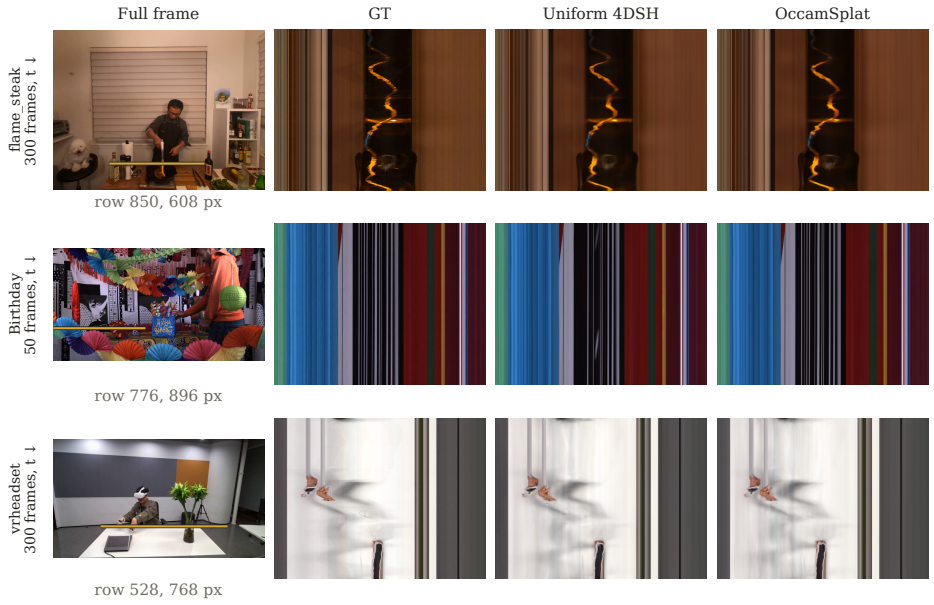

Figure 13: **Temporal coherence under selective appearance allocation.** The marked image row is stacked over test frames, with time increasing downward. Our renderings follow the ground-truth temporal structure and are about as coherent as uniform 4DSH.

D.3 CAPACITY AND IMAGE CONTRIBUTION

The learned allocation concentrates appearance capacity on Gaussians that contribute more to the image (Fig. 14). In the displayed examples, DC-only Gaussians make up much of the model but carry little compositing weight, while Gaussians with angular or temporal bands account for most of the visible content. Together with the visibility trend in Fig. 8, this pattern shows why primitive counts alone are an incomplete measure of appearance allocation.

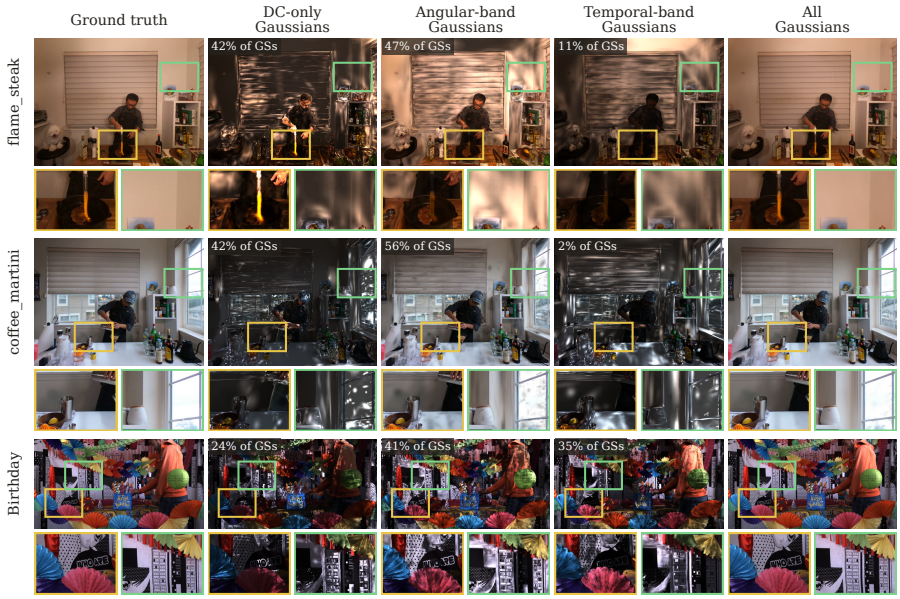

Figure 14: **Gaussians with higher capacity carry most visible content.** Panels group Gaussians by their highest active band and show the full render weighted by each group's per-pixel compositing share. Contrast is enhanced per panel for legibility; labels give the fraction of all Gaussians in each group.