Direct Conditional Parameterization for N-Dimensional Splatting

Zhongpai Gao* **Benjamin Planche** **Van Nguyen Nguyen**
Meng Zheng **Anwesa Choudhuri** **Arun Innanje**
Terrence Chen **Ziyan Wu**
United Imaging Intelligence, Boston, MA, USA

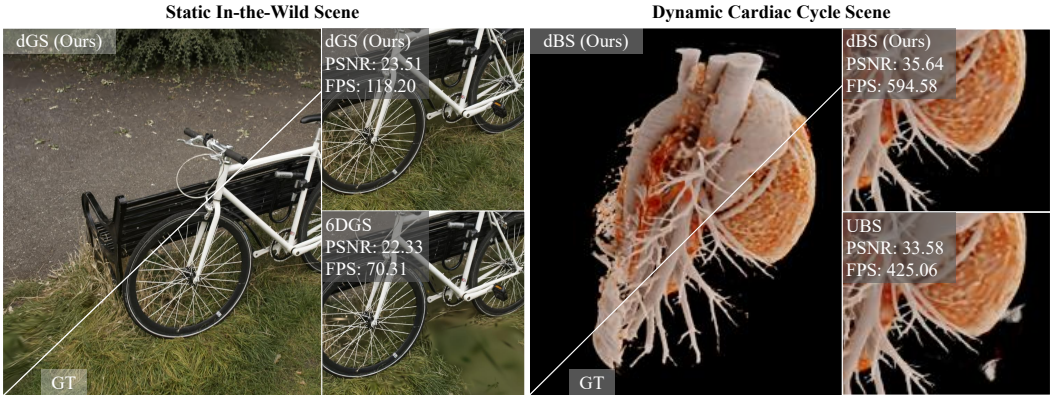


Figure 1: **Rendering quality and efficiency comparison.** Our direct conditional parameterization yields two methods: *d*GS (Gaussian kernel) outperforms its implicit-conditioning counterpart 6DGS [Gao et al., 2025a] on the static `bicycle` scene (left), and *d*BS (Beta kernel) outperforms UBS [Liu et al., 2026] on the dynamic `heart1` scene (right). Both achieve higher quality and higher efficiency than their covariance-derived predecessors.

Abstract

N-dimensional splatting extends 3D Gaussian Splatting with view direction and time to model view-dependent and dynamic effects. Each primitive is lifted into a joint distribution over position and conditioning variables, then conditionally sliced at each rendering query. N-DGS and UBS share this step across Gaussian and Beta kernels: recovering the conditional mean, opacity, and covariance requires per-primitive matrix inversions and regression products in every forward pass. We introduce direct conditional parameterization, replacing this covariance-derived readout with explicit per-effect parameters: a Cholesky precision factor for opacity and a factorized query-to-position regression operator with learnable per-dimension coupling. Unrestricted conditional coordinates recover a valid joint covariance; our factorization adds spatial-scale and coupling priors. Applied to both kernels, the parameterization yields *d*GS (Gaussian) and *d*BS (Beta). Across five static and dynamic benchmarks, they match or exceed their covariance-derived counterparts on quality, with gains up to +1.26 dB in the main MCMC evaluation. Production slicing is 6.9–7.7× faster than on-the-fly covariance-derived conditioning; complete-render profiling gives 1.25–1.91× speedups on four trained scenes. Direct conditioning provides a structured, kernel-agnostic alternative for N-dimensional splatting. Project page: https://gaozhongpai.github.io/dgs/.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

3D Gaussian Splatting (3DGS) [Kerbl et al., 2023] achieves photorealistic novel-view synthesis at real-time speeds through explicit 3D Gaussian primitives. Static primitives, however, struggle with specular reflections, temporal dynamics, and view-dependent geometry.

Recent work extends 3DGS along two orthogonal axes. The first is *kernel shape*: alternatives to the Gaussian kernel include Beta kernels [Liu et al., 2025a, 2026], generalized exponentials [Hamdi et al., 2024], and Student-t distributions [Zhu et al., 2025]. The second is *dimensionality*: N-dimensional Gaussians (N-DG) [Diolatzis et al., 2024], refined as 6DGS [Gao et al., 2025a] and 7DGS [Gao et al., 2025b] (collectively N-DGS), lift each primitive into a joint distribution over 3D position and conditioning variables (view direction with $C{=}3$, view+time with $C{=}4$). At render time, the $(3{+}C)$-dimensional primitive is *conditionally sliced* at the current query (a view direction or timestamp) using the conditional Gaussian formulas: the mean shifts, the opacity is modulated by a kernel evaluated at the query, and the spatial covariance receives a correction term. Universal Beta Splatting (UBS) [Liu et al., 2026] combines the two axes by replacing the Gaussian opacity decay with an anisotropic Beta kernel while inheriting the same conditional-slicing structure. N-DGS and UBS together represent the leading N-dimensional splatting formulations across Gaussian and Beta kernels.

Despite their different kernels, N-DGS and UBS share the same conditional-slicing step, and this is where the per-primitive cost concentrates. Read-out of the three effects requires inverting the query block, constructing a regression matrix, and applying a covariance-correction product (Table 1). These operations run once per primitive in every forward pass, so the total cost grows linearly with primitive count, which on large scenes like Mip-NeRF 360 reaches millions. Their ambient Cholesky coordinates also entangle the optimization of opacity precision, position regression, and conditional covariance, and conditioning can become numerically fragile when the query block is ill-conditioned.

Our insight. N-DGS uses ambient joint-covariance coordinates, although rendering needs only conditional spatial covariance, query precision, and query-to-position regression. Recovering them requires inversion and Schur complementation. Appendix C proves that this unrestricted operational tuple is a smooth, globally bijective coordinate chart for joint Gaussian covariances, with block-triangular reconstruction, an explicit inverse, and a nonvanishing coordinate-volume Jacobian throughout the SPD interior. *d*GS directly parameterizes a structured factorization of these rendering-aligned coordinates. It therefore both removes expensive operations and separates position from opacity coupling.

We introduce **direct conditional parameterization** for N-dimensional splatting. Opacity uses a directly parameterized Cholesky precision factor, and position uses a factorized regression operator with learnable per-dimension coupling that decouples position from opacity by default and recovers full coupling on demand. Because the change touches only the conditioning step, the same parameterization is kernel-agnostic and carries across kernel families: we instantiate it as *d*GS (Gaussian) for the direct counterpart of N-DGS and *d*BS (Beta) for the direct counterpart of UBS. Pairing each direct method with its covariance-derived counterpart (*d*GS vs. N-DGS, *d*BS vs. UBS) isolates the contribution of direct conditioning from the choice of kernel.

Our contributions are:

- **Direct conditional parameterization** for N-dimensional splatting: a kernel-agnostic alternative that explicitly parameterizes opacity precision and a factorized query-to-position regression operator. We prove that the underlying unrestricted operational tuple forms a smooth, globally bijective coordinate chart for joint Gaussian covariances, with an explicit inverse and a nonvanishing coordinate-volume Jacobian.
- **Two kernel instantiations and matched-pair evaluation.** We develop the formulation for the Gaussian kernel (*d*GS, the direct counterpart of N-DGS) and show that the same conditioning extends to the Beta kernel via a Beta-kernel instantiation (*d*BS, the direct counterpart of UBS). Pairing each direct method with its covariance-derived counterpart isolates the gain from direct conditioning from the choice of kernel.
- **Efficient CUDA implementation.** Fused, register-level CUDA kernels for direct conditioning and the N-DGS/UBS baselines achieve 6.9–7.7$\times$ faster production slicing than on-the-fly covariance-derived conditioning. Inference caching controls and complete-render profiling quantify the resulting speedups. The N-DGS baseline was previously only available as a PyTorch implementation; we provide an optimized CUDA comparison.

2 Related Work

3D Gaussian Splatting. 3D Gaussian Splatting (3DGS) [Kerbl et al., 2023] achieves real-time novel view synthesis through explicit 3D Gaussian primitives with spherical harmonic color encoding, and has inspired a wide range of extensions to anti-aliasing [Yu et al., 2024, Yan et al., 2024], densification [Kheradmand et al., 2024], compression [Lee et al., 2024, Morgenstern et al., 2024], and semantic modeling [Shi et al., 2024, Zhou et al., 2024]. The original formulation remains static, relying on spherical harmonics alone for view dependence and offering no mechanism for temporal dynamics or view-dependent geometry.

Alternative Kernel Designs. Several works replace the 3D Gaussian kernel with alternatives for better geometric fidelity: generalized exponentials [Hamdi et al., 2024], truncated [Liu et al., 2025b], second-order [Zhang et al., 2025], half-Gaussian [Li et al., 2025], and discontinuity-aware [Qu et al., 2024] variants; Beta [Liu et al., 2025a], Student-t [Zhu et al., 2025], and Gabor [Zhou et al., 2025] distributions; and alternative primitives such as convex shapes [Held et al., 2025], triangles [Held et al., 2026], and radiance meshes [Mai et al., 2026]. Orthogonal to kernel shape, Textured Gaussians [Chao et al., 2025] models spatially varying color and opacity, while Spherical Voronoi [Di Sario et al., 2026] provides a directional appearance representation. These works modify primitives or appearance models rather than replacing covariance-derived conditional slicing; UBS combines the Beta kernel with N-dimensional conditioning, discussed next.

N-Dimensional Splatting. To model view-dependent and temporal effects, recent methods lift primitives into higher-dimensional spaces, conditioning on view direction or time alongside position. 4DGS [Yang et al., 2024] adds a temporal dimension for dynamic scene reconstruction, and N-DG [Diolatzis et al., 2024] establishes a general framework for arbitrary-dimensional Gaussians. 6DGS [Gao et al., 2025a] jointly models spatial position and view direction; 7DGS [Gao et al., 2025b] adds time as well; we refer to the two collectively as N-DGS. Universal Beta Splatting (UBS) [Liu et al., 2026] extends the same paradigm to Beta kernels with per-dimension bandwidth control. N-DGS and UBS together cover the two kernel families currently used in N-dimensional splatting, and both inherit the covariance-derived conditioning step that our work replaces.

Dynamic Scene Modeling. NeRF-based methods model dynamics through deformations: D-NeRF [Pumarola et al., 2021] learns a canonical scene and a time-dependent deformation field, Nerfies [Park et al., 2021a] handles non-rigid motion, and HyperNeRF [Park et al., 2021b] captures topological changes. For Gaussian Splatting, 4DGS [Yang et al., 2024] slices native spatiotemporal primitives for real-time rendering. Dynamic 3D Gaussians [Luiten et al., 2024] optimizes persistent primitives' motion and rotation under local-rigidity constraints, while FreeTimeGS [Wang et al., 2025] allows Gaussians to appear at arbitrary times and locations with motion functions. Tracked and motion-based representations evolve 3D primitives explicitly; N-dimensional splatting instead obtains query-dependent primitives by conditional slicing.

3 Preliminary

3D Gaussian Splatting 3DGS [Kerbl et al., 2023] represents a scene as a collection of anisotropic 3D Gaussians. Each primitive carries a mean $\boldsymbol{\mu} \in \mathbb{R}^3$, a covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$ factored as $\boldsymbol{\Sigma} = \mathbf{RSS}^\top\mathbf{R}^\top$ with rotation $\mathbf{R}$ (quaternion) and diagonal scale $\mathbf{S} = \mathrm{diag}(s_x, s_y, s_z)$, an opacity $\alpha \in [0, 1]$, and view-dependent color via spherical harmonics. At render time, primitives are projected to 2D screen-space Gaussians, sorted by depth per tile, and alpha-composited front-to-back to produce the final pixel color. The per-pixel contribution of each primitive is $\alpha \cdot \mathcal{K}_G(\boldsymbol{\delta}; \boldsymbol{\Sigma}_{\text{2D}}) = \alpha \cdot \exp\big(-\frac{1}{2}\boldsymbol{\delta}^\top\boldsymbol{\Sigma}_{\text{2D}}^{-1}\boldsymbol{\delta}\big)$, where $\boldsymbol{\delta}$ is the offset from the pixel to the projected primitive center, $\boldsymbol{\Sigma}_{\text{2D}}$ is the projected 2D covariance, and $\mathcal{K}_G$ denotes the Gaussian kernel. The representation is static: it has no mechanism for temporal dynamics or view-dependent geometry.

N-Dimensional Splatting To capture view-dependent and temporal effects, N-DGS [Gao et al., 2025a,b] (Gaussian kernel) and UBS [Liu et al., 2026] (Beta kernel) parameterize each primitive as a joint distribution over position $\mathbf{X}_p \in \mathbb{R}^3$ and query variables $\mathbf{X}_q \in \mathbb{R}^C$ (view direction for $C = 3$, view+time for $C = 4$):

$$\begin{pmatrix}\mathbf{X}_p\\ \mathbf{X}_q\end{pmatrix} \sim \left(\begin{pmatrix}\boldsymbol{\mu}_p\\ \boldsymbol{\mu}_q\end{pmatrix}, \begin{pmatrix}\boldsymbol{\Sigma}_{pp} & \boldsymbol{\Sigma}_{pq}\\ \boldsymbol{\Sigma}_{pq}^\top & \boldsymbol{\Sigma}_{qq}\end{pmatrix}\right). \tag{1}$$

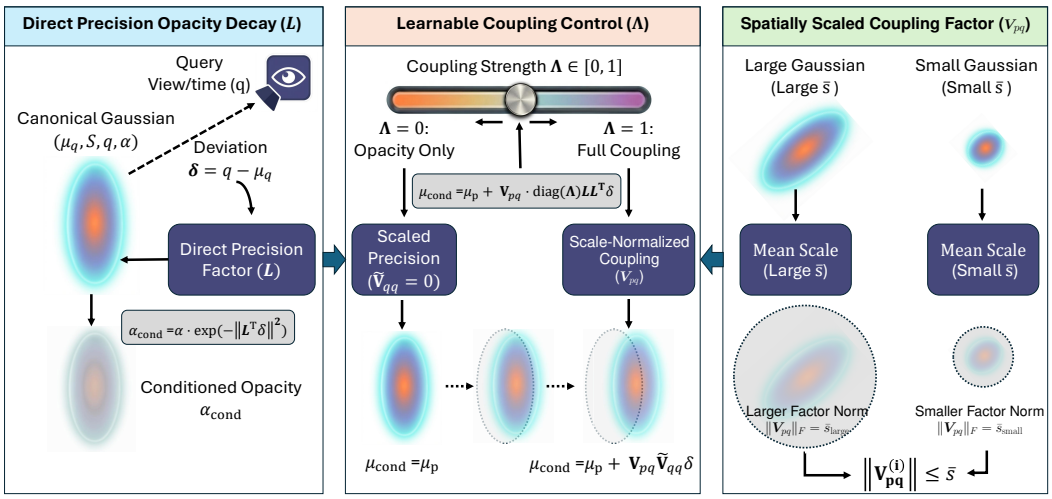


Figure 2: Overview of the *d*GS pipeline. Given query deviation $\boldsymbol{\delta} = \mathbf{q} - \boldsymbol{\mu}_q$, opacity uses Cholesky factor $\mathbf{L}$, while position uses the scale-normalized coupling factor $\mathbf{V}_{pq}$ and scaled precision $\tilde{\mathbf{V}}_{qq}$ with per-dimension coupling $\boldsymbol{\Lambda}$. The factor norm follows the primitive scale; the complete regression and final shift also depend on $\tilde{\mathbf{V}}_{qq}$ and $\boldsymbol{\delta}$. The same pipeline extends to *d*BS (Section 4.4).

At query $\mathbf{q}$, N-DGS derives conditional position, spatial covariance, and opacity as

$$\boldsymbol{\mu}_{\text{cond}} = \boldsymbol{\mu}_p + \boldsymbol{\Sigma}_{pq}\,\boldsymbol{\Sigma}_{qq}^{-1}(\mathbf{q} - \boldsymbol{\mu}_q), \tag{2}$$

$$\boldsymbol{\Sigma}_{\text{cond}} = \boldsymbol{\Sigma}_{pp} - \boldsymbol{\Sigma}_{pq}\,\boldsymbol{\Sigma}_{qq}^{-1}\,\boldsymbol{\Sigma}_{pq}^{\top}, \tag{3}$$

$$\alpha_{\text{cond}} = \alpha \cdot \mathcal{K}\big(\mathbf{q} - \boldsymbol{\mu}_q;\, \boldsymbol{\Sigma}_{qq}\big)\,. \tag{4}$$

$\mathcal{K}(\cdot; \boldsymbol{\Sigma}_{qq})$ is evaluated through the query precision $\boldsymbol{\Sigma}_{qq}^{-1}$. N-DGS uses $\mathcal{K}_G(\boldsymbol{\delta}; \boldsymbol{\Sigma}_{qq}) = \exp(-\lambda\,\boldsymbol{\delta}^{\top}\boldsymbol{\Sigma}_{qq}^{-1}\boldsymbol{\delta})$ with scaling λ (default 0.35). UBS retains the covariance-derived structure with a Beta opacity kernel and bandwidth-dependent mean/covariance coupling (Appendix F). Both require query-block inversion, regression construction, and covariance correction; our direct formulation exposes the rendered quantities instead.

4 Our Approach

We replace the covariance-derived conditioning step with three explicit parameters per primitive: a Cholesky factor $\mathbf{L}$ for opacity precision, a spatial coupling factor $\mathbf{V}_{pq}$ for position, and a per-dimension coupling vector $\boldsymbol{\Lambda}$ that ties the two. We develop the formulation for the Gaussian kernel below, yielding *d*GS, and show in Section 4.4 that swapping in the Beta opacity decay yields *d*BS.

4.1 Overview

Given the deviation $\boldsymbol{\delta} = \mathbf{q} - \boldsymbol{\mu}_q$ from the canonical view/time, *d*GS replaces Equations (2) and (4) with two directly parameterized components:

$$\alpha_{\text{cond}} = \alpha \cdot \exp\big(-\lambda_o\,\|\mathbf{L}^{\top}\boldsymbol{\delta}\|_2^2\big)\,, \qquad \text{(Opacity)} \tag{5}$$

$$\boldsymbol{\mu}_{\text{cond}} = \boldsymbol{\mu}_p + \mathbf{V}_{pq}\,\text{diag}(\boldsymbol{\Lambda})\,\mathbf{V}_{qq}\,\boldsymbol{\delta}. \qquad \text{(Position)} \tag{6}$$

The parameterization is defined by three learned quantities ($\mathbf{L}$, $\mathbf{V}_{pq}$, $\boldsymbol{\Lambda}$) and one fixed constant (λ_o=0.35, the opacity scaling factor inherited from N-DGS). $\mathbf{L} \in \mathbb{R}^{C\times C}$ is a lower-triangular Cholesky factor with an exponential activation on the diagonal, so that $\mathbf{V}_{qq} = \mathbf{L}\mathbf{L}^{\top}$ is symmetric positive-definite by construction and *is* the precision matrix without any inversion. $\mathbf{V}_{pq} = \bar{s}\boldsymbol{\Theta}_{pq} \in \mathbb{R}^{3\times C}$ is a spatial coupling factor, written as the average spatial scale $\bar{s} = \text{mean}(s_x, s_y, s_z)$ times a learnable direction $\boldsymbol{\Theta}_{pq}$ with unit Frobenius norm. This gives $\|\mathbf{V}_{pq}\|_F = \bar{s}$; the effective regression also depends on $\text{diag}(\boldsymbol{\Lambda})\mathbf{V}_{qq}$. $\boldsymbol{\Lambda} \in [0, 1]^C$ weights the precision-transformed query channels in the position update: at $\boldsymbol{\Lambda} = \mathbf{0}$ position decouples entirely (recovering an opacity-only variant we

Table 1: Per-primitive operations and production slicing-only forward time on A100 (1M primitives). Direct methods avoid query-block inversion, regression construction, and covariance correction. Parentheses compare optimized on-the-fly conditioning; Appendix H reports separate cached and factorized inference controls.

	Implicit		Direct (ours)	
Operation	**N-DGS**	**UBS**	***d*GS**	***d*BS**
Invert $\mathbf{\Sigma}_{qq}$	$\mathbf{\Sigma}_{qq}^{-1}$	$\mathbf{\Sigma}_{qq}^{-1}$	—	—
Regr. matrix $\mathbf{P}$	$\mathbf{\Sigma}_{pq}\mathbf{\Sigma}_{qq}^{-1}$	$\mathbf{\Sigma}_{pq}\mathbf{\Sigma}_{qq}^{-1}\mathrm{diag}(\mathbf{\Lambda}_\beta)$	—	—
Cov. correction	$\mathbf{P}\,\mathbf{\Sigma}_{pq}^\top$	$\mathbf{P}\,\mathbf{\Sigma}_{pq}^\top$	—	—
Position shift	$\mathbf{P}\,\boldsymbol{\delta}$	$\mathbf{P}\,\boldsymbol{\delta}$	$\mathbf{V}_{pq}\mathrm{diag}(\mathbf{\Lambda})\mathbf{V}_{qq}\boldsymbol{\delta}$	$\mathbf{V}_{pq}\mathrm{diag}(\mathbf{\Lambda}_\beta)\mathbf{V}_{qq}\boldsymbol{\delta}$
Opacity	$\mathcal{K}_G(\boldsymbol{\delta}^\top\mathbf{\Sigma}_{qq}^{-1}\boldsymbol{\delta})$	$\mathcal{K}_B(\hat{\mathbf{L}}_q^{-1}\boldsymbol{\delta};\boldsymbol{\beta})$	$\mathcal{K}_G(\|\mathbf{L}^\top\boldsymbol{\delta}\|^2)$	$\mathcal{K}_B(\mathbf{L}^\top\boldsymbol{\delta};\boldsymbol{\beta})$
Slicing time, C=3 (ms)	0.71	0.67	**0.10** (**7.1**×)	**0.10** (**6.9**×)
Slicing time, C=4 (ms)	0.96	1.34	**0.12** (**7.7**×)	**0.13** (**7.2**×)

denote *d*GS-O, with $\boldsymbol{\mu}_{\text{cond}} = \boldsymbol{\mu}_p$), at $\mathbf{\Lambda} = \mathbf{1}$ it couples through the full opacity precision $\mathbf{V}_{qq}$, and intermediate values continuously weight these channels. Channels of the same type (*e.g.*, the three view dimensions) share a logit.

The rendered conditional covariance $\mathbf{\Sigma}_{\text{cond}} = \mathbf{R}\mathbf{S}\mathbf{S}^\top\mathbf{R}^\top$ remains query-independent and unchanged from 3DGS, preserving compatibility with existing densification strategies. It differs from the spatial marginal of a reconstructed joint Gaussian: Appendix C gives the exact correspondence for unrestricted conditional coordinates. The implemented factorization constrains the regression through spatial normalization and bounded coupling, changing its optimization geometry and priors.

Table 1 contrasts the per-primitive operations: N-DGS constructs $\mathbf{P} = \mathbf{\Sigma}_{pq}\mathbf{\Sigma}_{qq}^{-1}$ through query-block inversion and computes a covariance correction; UBS additionally applies bandwidth-dependent coupling. Direct methods apply their regression operator without deriving these quantities. The coupling $\mathbf{\Lambda}$ adds explicit control: $\mathbf{\Lambda} = \mathbf{1}$ uses the full direct precision path and $\mathbf{\Lambda} = \mathbf{0}$ recovers *d*GS-O. Intermediate values let each query type learn a partial coupling strength.

4.2 Opacity Conditioning

The Cholesky factor $\mathbf{L}$ is parameterized directly, with exponential activation on the diagonal so that $\mathbf{V}_{qq} = \mathbf{L}\mathbf{L}^\top$ is symmetric positive-definite. Evaluating opacity reduces to a triangular matrix-vector product $\mathbf{L}^\top\boldsymbol{\delta}$ followed by a squared norm, avoiding query-block inversion.

For $C = 4$ (view+time), all ten lower-triangular entries of $\mathbf{L}$ are learned. View–time off-diagonals start at zero but remain unconstrained. With $\boldsymbol{\delta} = (\boldsymbol{\delta}_v, \delta_t)^\top$, opacity is

$$\alpha_{\text{cond}} = \alpha \cdot \exp\Big(-\lambda_o \left(\boldsymbol{\delta}_v^\top \mathbf{V}_{vv}\boldsymbol{\delta}_v + 2\delta_t\boldsymbol{\delta}_v^\top\mathbf{v}_{vt} + \delta_t^2 v_t\right)\Big), \tag{7}$$

where $\mathbf{V}_{vv}$, $\mathbf{v}_{vt}$, and v_t are the view, cross, and time blocks of the full precision $\mathbf{V}_{qq}$.

4.3 Position Conditioning

The position update (6) factors into the spatial coupling factor $\mathbf{V}_{pq}$, the per-dimension coupling $\mathrm{diag}(\mathbf{\Lambda})$, and the precision $\mathbf{V}_{qq}$ shared with opacity:

$$\boldsymbol{\mu}_{\text{cond}} = \boldsymbol{\mu}_p + \underbrace{\bar{s}\,\mathbf{\Theta}_{pq}}_{\mathbf{V}_{pq}} \underbrace{\mathrm{diag}(\mathbf{\Lambda})\,\mathbf{V}_{qq}}_{\tilde{\mathbf{V}}_{qq}}\,\boldsymbol{\delta}.$$

The coupling $\mathbf{\Lambda}$ interpolates between two extremes per primitive: at $\mathbf{\Lambda} = \mathbf{0}$, position is query-independent (recovering *d*GS-O); at $\mathbf{\Lambda} = \mathbf{1}$, position and opacity share the full precision $\mathbf{V}_{qq}$, mirroring N-DGS's coupling structure. Precision-transformed query channels can learn different coupling strengths.

With $\mathbf{\Lambda} = \mathbf{0}$, opacity conditioning remains active under Equation (5). A primitive can therefore be suppressed under an unfavorable view without moving its center; stronger coupling permits query-dependent motion with the same base spatial covariance.

Spatial scaling. We set $\mathbf{V}_{pq} = \bar{s}\,\mathbf{\Theta}_{pq}$ with $\|\mathbf{\Theta}_{pq}\|_F = 1$, giving $\|\mathbf{V}_{pq}\|_F = \bar{s}$ and $\|\mathbf{V}_{pq}^{(i)}\|_2 \le \bar{s}$. This is an explicit spatial-scale prior on the coupling factor, not a bound on the final shift: the effective

regression is $\mathbf{M}_{\mathrm{dGS}} = \mathbf{V}_{pq}\,\mathrm{diag}(\mathbf{\Lambda})\mathbf{V}_{qq}$ and the shift also depends on the query deviation. N-DGS instead obtains its regression operator indirectly from shared covariance blocks. Appendix D derives both forms and a basis-independent Gaussian bound; the ablation in Table 3 confirms the value of the prior, as removing $\bar{s}$ degrades PSNR by 2.65 dB on NeRF Synthetic. The mean scale follows primitive size symmetrically across principal axes, while query precision and $\mathbf{\Lambda}$ retain separate control of opacity and position coupling.

4.4 Extension to Beta Kernel: *d*BS

Direct conditioning touches only the slicing step (precision read-out, regression, and coupling), making it kernel-agnostic: any opacity decay that consumes the precision-Cholesky distance $\mathbf{L}^\top\boldsymbol{\delta}$ can be substituted in. We instantiate this with a Beta opacity decay following UBS [Liu et al., 2026] to obtain *d*BS (direct Beta Splatting), the direct counterpart of UBS.

We follow the released UBS implementation used in our experiments. UBS evaluates opacity as $\alpha\prod_i(1-\tanh((\hat{\mathbf{L}}_q^{-1}\boldsymbol{\delta})_i^2))^{\beta_i}$, with $\hat{\mathbf{L}}_q\hat{\mathbf{L}}_q^\top = \mathbf{\Sigma}_{qq}$. *d*BS replaces this triangular solve with the direct product $\mathbf{z} = \mathbf{L}^\top\boldsymbol{\delta}$ while retaining the same bandwidth activation and coupling convention:

$$\alpha_{\text{cond}} = \alpha \cdot \prod_{i=1}^{C}\left(1-\tanh(z_i^2)\right)^{\beta_i}, \tag{8}$$

$$\boldsymbol{\mu}_{\text{cond}} = \boldsymbol{\mu}_p + \mathbf{V}_{pq}\,\mathrm{diag}(\mathbf{\Lambda}_\beta)\,\mathbf{V}_{qq}\,\boldsymbol{\delta}, \tag{9}$$

where $\beta_i = 4\exp(\beta_i^{\text{raw}})$ and $\mathbf{\Lambda}_\beta = \min(\boldsymbol{\beta}/4, \mathbf{1})$, as in UBS. The position update reuses the opacity bandwidth for coupling instead of a separate learned $\mathbf{\Lambda}$. The precision factor, spatial factor, and directly stored conditional covariance transfer from *d*GS. Including the spatial Beta parameter, UBS and *d*BS each use 19/27 conditioning parameters for $C = 3/4$; N-DGS and *d*GS use 15/16 and 22/24, respectively. Appendix F gives the source-aligned UBS formulation.

The shared bandwidth gives opacity and position a consistent response to the query. Dividing $\boldsymbol{\beta}$ by four puts the coupling near unit magnitude when its raw parameter is near zero, while clipping prevents a larger bandwidth from increasing the coupling beyond one. Gaussian and Beta instances thus retain the same direct conditional structure, with a separate learned coupling in *d*GS and a bandwidth-derived coupling in *d*BS.

4.5 CUDA Implementation

The original N-DGS methods (6DGS [Gao et al., 2025a], 7DGS [Gao et al., 2025b]) rely on PyTorch's autograd for the conditional Gaussian operations, so for a fair comparison we implement *d*GS and the N-DGS baseline with custom CUDA kernels. The kernel fuses the conditional position and opacity computation in a single pass (Algorithm 1 in Appendix B), and *d*BS reuses the same pipeline with the opacity decay swapped (Algorithm 2).

Kernel design. The kernel is templated on C (3 or 4) with compile-time loop unrolling, and each thread processes one primitive. The triangular product $\mathbf{L}^\top\boldsymbol{\delta}$ uses only $C(C+1)/2$ multiplications followed by a dot product, all in per-thread registers; the small $3\times C$ and $C\times C$ matrices fit entirely in registers, so the kernel allocates no intermediate tensors and avoids shared-memory traffic. The N-DGS baseline also fuses inversion, regression construction, and Schur correction, using a cofactor inverse for $C = 3$ and register-local Gauss–Jordan elimination for $C = 4$.

Kernel benchmarks. Table 1 reports production CUDA slicing at 1M primitives on an A100, excluding autograd and rasterization. Direct conditioning is **6.9–7.7× faster** than optimized on-the-fly conditioning. In a separate harness with materialized inputs, *d*GS remains 2.0–2.3× faster than inference-cached N-DGS and 1.8× faster than a factorized-query control (Appendix H). Profiling four trained scenes gives 1.25–1.91× complete-render gains: shared rasterizer code still processes different learned footprints and tile workloads, limiting the benefit of faster slicing.



Figure 3: Qualitative comparison on static and dynamic scenes. Insets use identical crop coordinates and magnification across methods; boxes mark the source regions.

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate the *d*-splatting family across five diverse datasets covering both static and dynamic scenes:

- *Static scenes:* (1) *NeRF Synthetic* [Mildenhall et al., 2020] with 8 synthetic scenes under controlled lighting for evaluating geometric reconstruction; (2) *Mip-NeRF 360* [Barron et al., 2022] with 9 challenging unbounded real-world scenes for evaluating view synthesis; and (3) *6DGS-PBR* [Gao et al., 2025a] featuring 7 physically-based scenes with complex view-dependent effects.
- *Dynamic scenes:* (4) *D-NeRF* [Pumarola et al., 2021] with 8 synthetic sequences featuring various deformation types; and (5) *7DGS-PBR* [Gao et al., 2025b] containing 6 physically-based dynamic scenes with complex spatio-temporal-angular correlations.

Evaluation Metrics. We assess reconstruction quality using Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) [Wang et al., 2004], and Learned Perceptual Image Patch Similarity (LPIPS) [Zhang et al., 2018]. We follow the 3DGS [Kerbl et al., 2023] evaluation protocol (joint-RGB PSNR, white background on NeRF Synthetic). We re-evaluate UBS [Liu et al., 2026] under this protocol; its released evaluator instead averages channel-wise PSNR and uses a black background on NeRF Synthetic. We also report training time (minutes), Gaussian count (K), and two rendering speeds: FPS uses a PyTorch reference implementation for slicing, while FPS_{CUDA} uses the fused CUDA slicing kernels.

Table 2: MCMC results: scene means from one run per scene (#Gauss in K, training time in minutes; dynamic batch size 1). Pairs share per-scene primitive caps; realized counts are reported. FPS and FPS_{CUDA} are arithmetic scene means using PyTorch and fused CUDA slicing.

	Dataset	Method	#Gauss	Time ↓	FPS ↑	FPS_{CUDA} ↑	PSNR ↑	SSIM ↑	LPIPS ↓
STATIC	NeRF Synthetic	N-DGS	300	10.43	183.58	348.95	34.04	0.973	0.027
		*d*GS (**Ours**)	300	**10.26**	**245.18**	**593.94**	**34.34**	**0.973**	**0.026**
		UBS	300	6.80	155.96	306.33	34.48	0.975	0.026
		*d*BS (**Ours**)	300	**5.48**	**239.09**	**415.66**	**34.53**	**0.975**	**0.025**
	Mip-NeRF 360	N-DGS	2,852	51.23	35.69	75.31	26.88	0.775	0.263
		*d*GS (**Ours**)	2,574	**49.85**	**42.49**	**200.56**	**27.67**	**0.794**	**0.227**
		UBS	3,057	28.59	40.94	76.18	28.50	0.841	**0.183**
		*d*BS (**Ours**)	3,057	**24.42**	**62.05**	**157.45**	**28.70**	**0.842**	0.185
	6DGS-PBR	N-DGS	163	54.80	328.75	476.79	40.06	0.976	0.039
		*d*GS (**Ours**)	163	**47.54**	**356.15**	**551.02**	**40.34**	**0.976**	**0.038**
		UBS	163	8.70	239.87	348.63	39.53	0.975	0.042
		*d*BS (**Ours**)	163	**8.47**	**305.33**	**401.93**	**39.72**	**0.976**	**0.042**
DYNAMIC	D-NeRF	N-DGS	150	**10.31**	306.51	501.52	34.63	0.976	0.033
		*d*GS (**Ours**)	150	10.61	**407.40**	**746.65**	**34.64**	**0.976**	**0.030**
		UBS	150	**8.37**	229.83	371.81	32.91	0.967	0.040
		*d*BS (**Ours**)	150	9.34	**323.75**	**463.09**	**33.55**	**0.972**	**0.033**
	7DGS-PBR	N-DGS	175	**28.44**	294.48	511.12	29.20	0.944	0.063
		*d*GS (**Ours**)	175	31.12	**366.06**	**577.38**	**29.90**	**0.948**	**0.061**
		UBS	175	9.90	232.17	396.33	30.61	0.951	0.057
		*d*BS (**Ours**)	175	**8.79**	**347.93**	**539.57**	**31.87**	**0.955**	**0.053**

Implementation Details. All methods are implemented in PyTorch with custom CUDA kernels on a single NVIDIA A100 GPU and rendered with the TC-GS rasterizer [Liao et al., 2025]. We compare two head-to-head pairs (*d*GS vs. N-DGS, *d*BS vs. UBS) and additionally include *d*GS-O (*d*GS with $\mathbf{\Lambda}=\mathbf{0}$) as a maximum-efficiency variant. All methods share the same training pipeline: Adam optimizer, 30K iterations, identical learning rates and densification schedules per dataset. Main dynamic MCMC results use batch size 1 for both methods; Appendix G checks the Beta pair at the original batch size 4. Main benchmark entries are single-run scene averages; repeated-seed controls are reported separately. For *d*GS and *d*BS, the view mean $\boldsymbol{\mu}_q$ is initialized to random unit vectors (with time uniformly sampled in $[0, 1]$ for dynamic scenes), the coupling-direction parameters $\mathbf{\Theta}_{pq}$ from $\mathcal{N}(0, 0.01^2)$, and the Cholesky factor $\mathbf{L}$ with diagonal elements $\log(2.0)$ and zero off-diagonals.

5.2 Results with MCMC Densification

Table 2 compares each direct method against its covariance-derived counterpart under MCMC densification [Kheradmand et al., 2024]. Pairs share per-scene primitive caps, while realized counts can differ. We cap primitives at 300K for NeRF Synthetic and 150K for D-NeRF, with scene-specific caps for Mip-NeRF 360 (1.5M–6M), 6DGS-PBR (60K–300K), and 7DGS-PBR (150K–300K).

Gaussian-kernel pair (*d*GS vs. N-DGS). *d*GS matches or improves dataset-mean quality while increasing rendering throughput. On Mip-NeRF 360, it gains +0.79 dB with 2.574M versus 2.852M primitives, approximately 10% fewer. Arithmetic-mean FPS_{CUDA} is 200.56 versus 75.31 (a 2.66× ratio); this aggregation differs from mean frame time, and realized counts also affect speed. NeRF Synthetic gains +0.30 dB and +70% mean FPS (593.94 vs. 348.95). D-NeRF gives quality parity (+0.01 dB) and +49% mean FPS (746.65 vs. 501.52). On 7DGS-PBR, which combines dynamics with view-dependent specular effects, *d*GS gains +0.70 dB and +13% mean FPS (577.38 vs. 511.12).

Beta-kernel pair (*d*BS vs. UBS). The same throughput pattern transfers to the Beta kernel. On NeRF Synthetic, *d*BS gives quality parity (+0.05 dB) while rendering +36% faster (415.66 vs. 306.33) and training in 5.48 vs. 6.80 minutes. On Mip-NeRF 360, *d*BS improves PSNR by +0.20 dB and renders +107% faster (157.45 vs. 76.18). On D-NeRF, *d*BS gains +0.64 dB with +25% faster rendering (463.09 vs. 371.81). The 7DGS-PBR gap is largest: +1.26 dB and +36% (539.57 vs. 396.33), with per-scene PSNR gains reaching +2.06 dB on `heart`. These results support computational gains across kernels; quality depends on the full parameterization and its optimization priors.

Table 3: Ablation study on NeRF Synthetic and 6DGS-PBR. Λ=0 gives *d*GS-O (opacity only), Λ=1 fixes full coupling, and learned Λ gives full *d*GS. "w/o $\bar{s}$" removes spatial scaling.

Config	NeRF Synthetic			6DGS-PBR		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Λ=0 (*d*GS-O)	33.84	0.971	0.030	36.54	0.965	0.059
Λ=1 (fixed)	33.78	0.971	0.029	37.31	**0.972**	0.053
*d*GS w/o $\bar{s}$	31.26	0.957	0.046	34.78	0.951	0.075
Λ learned (*d*GS)	**33.91**	**0.972**	**0.029**	**37.75**	**0.972**	**0.052**

Slicing and complete-render cost. The two FPS columns compare PyTorch and fused CUDA slicing within each method. On Mip-NeRF 360, this increases N-DGS mean FPS by 2.11× and *d*GS mean FPS by 4.72×. On four profiled checkpoints, slicing occupies 42–64% of baseline frame time; projection, binning, sorting, and rasterization account for the remainder. Appendix H separates these costs and explains why shared downstream code does not imply equal rendering workload.

Figure 3 shows qualitative comparisons. On the three static scenes, *d*GS, UBS, and *d*BS all produce visually clean reconstructions, while *d*GS shows noticeably fewer artifacts than N-DGS. On the two dynamic scenes, *d*GS and *d*BS both deliver better visual quality than their covariance-derived counterparts (N-DGS and UBS, respectively). Standard-densification results for *d*GS vs. N-DGS, the protocol N-DGS originally evaluates with, appear in Appendix E. A broader comparison against SOTA static-scene methods on Mip-NeRF 360 and NeRF Synthetic, where *d*BS achieves the best PSNR on Mip-NeRF 360 and ties for best PSNR on NeRF Synthetic, is provided in Appendix A.

5.3 Ablation Study

Table 3 ablates the two key design choices: the per-dimension coupling $\mathbf{\Lambda}$ and the spatial scaling $\bar{s}$ in the coupling factor $\mathbf{V}_{pq}$. All ablation runs use standard 3DGS densification [Kerbl et al., 2023] on NeRF Synthetic and 6DGS-PBR (the same protocol as the appendix Standard-Densification results). An unrestricted direct-regression control reaches 32.91/36.44 dB versus *d*GS's 33.91/37.75 dB, supporting the role of structure and optimization priors beyond direct readout (Appendix G).

Learned $\mathbf{\Lambda}$ outperforms both fixed extremes. On NeRF Synthetic, learned $\mathbf{\Lambda}$ achieves 33.91 dB vs. 33.84 ($\mathbf{\Lambda}$=$\mathbf{0}$, opacity-only) and 33.78 ($\mathbf{\Lambda}$=$\mathbf{1}$, full coupling). The gap widens on 6DGS-PBR (view-dependent specular scenes): 37.75 dB vs. 36.54 ($\mathbf{\Lambda}$=$\mathbf{0}$, -1.21 dB) and 37.31 ($\mathbf{\Lambda}$=$\mathbf{1}$, -0.44 dB). These results suggest that the optimal coupling is scene-dependent and may be primitive-dependent: the diffuse NeRF Synthetic scenes prefer weak coupling ($\mathbf{\Lambda}$=$\mathbf{0}$ beats $\mathbf{\Lambda}$=$\mathbf{1}$), the specular 6DGS-PBR scenes prefer strong coupling ($\mathbf{\Lambda}$=$\mathbf{1}$ beats $\mathbf{\Lambda}$=$\mathbf{0}$ by +0.77 dB), and learned per-Gaussian $\mathbf{\Lambda}$ improves on both extremes on both datasets.

Spatial scaling is critical. Removing $\bar{s}$ degrades PSNR by 2.65 dB on NeRF Synthetic (33.91 vs. 31.26) and 2.97 dB on 6DGS-PBR (37.75 vs. 34.78). Notably, an unscaled coupling factor is worse than no position regression at all (the $\mathbf{\Lambda}$=$\mathbf{0}$ baseline at 33.84 / 36.54 beats *d*GS w/o $\bar{s}$ at 31.26 / 34.78). This supports the spatial-scale prior on $\mathbf{V}_{pq}$ analyzed in Appendix D; the effective regression and final shift additionally depend on $\mathrm{diag}(\mathbf{\Lambda})\mathbf{V}_{qq}$ and the query deviation.

Scope and limitations. Our evaluation covers five static and dynamic benchmarks; other kernels and city-scale scenes require separate validation of memory use and visibility handling.

6 Conclusion

We presented direct conditional parameterization for N-dimensional splatting. Whereas N-DGS derives rendered quantities from a joint covariance through inversion and Schur complementation, *d*GS directly parameterizes conditional quantities with explicit spatial-scale and coupling priors. Unrestricted conditional coordinates preserve the joint Gaussian covariance family, while the structured factorization makes rendering parameters explicit and gives optimization direct control over their coupling. Across five benchmarks and two kernels, the resulting methods match or improve aggregate quality. Production slicing is 6.9–7.7× faster than on-the-fly conditioning, with smaller complete-render gains determined by downstream workload. Future work can extend the formulation to other primitive attributes, conditioning queries, and kernel families.

Funding and competing interests. This work was supported by United Imaging Intelligence through employment and institutional computing resources. All authors are affiliated with the company and report no additional third-party funding or competing financial interests.

References

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *CVPR*, 2022. 7, 13, 18

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. In *ICCV*, 2023. 13

Brian Chao, Hung-Yu Tseng, Lorenzo Porzi, Chen Gao, Tuotuo Li, Qinbo Li, Ayush Saraf, Jia-Bin Huang, Johannes Kopf, Gordon Wetzstein, and Changil Kim. Textured Gaussians for Enhanced 3D Scene Appearance Modeling. In *CVPR*, 2025. 3, 13

Francesco Di Sario, Daniel Rebain, Dor Verbin, Marco Grangetto, and Andrea Tagliasacchi. Spherical Voronoi: Directional Appearance as a Differentiable Partition of the Sphere. In *CVPR*, pages 22529–22538, 2026. 3, 13

Stavros Diolatzis, Tobias Zirr, Alexandr Kuznetsov, Georgios Kopanas, and Anton Kaplanyan. N-Dimensional Gaussians for Fitting of High Dimensional Functions. In *SIGGRAPH*, 2024. 2, 3

Zhongpai Gao, Benjamin Planche, Meng Zheng, Anwesa Choudhuri, Terrence Chen, and Ziyan Wu. 6DGS: Enhanced Direction-Aware Gaussian Splatting for Volumetric Rendering. In *ICLR*, 2025a. 1, 2, 3, 6, 7, 16, 18

Zhongpai Gao, Benjamin Planche, Meng Zheng, Anwesa Choudhuri, Terrence Chen, and Ziyan Wu. 7DGS: Unified Spatial-Temporal-Angular Gaussian Splatting. In *ICCV*, 2025b. 2, 3, 6, 7, 16, 18

Abdullah Hamdi, Luke Melas-Kyriazi, Jinjie Mai, Guocheng Qian, Ruoshi Liu, Carl Vondrick, Bernard Ghanem, and Andrea Vedaldi. GES: Generalized Exponential Splatting for Efficient Radiance Field Rendering. In *CVPR*, 2024. 2, 3, 13

Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Deliege, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3D Convex Splatting: Radiance Field Rendering with 3D Smooth Convexes. In *CVPR*, 2025. 3

Jan Held, Renaud Vandeghen, Adrien Deliege, Abdullah Hamdi, Silvio Giancola, Daniel Rebain, Anthony Cioppa, Bernard Ghanem, Andrea Vedaldi, Andrea Tagliasacchi, and Marc Van Droogenbroeck. Triangle Splatting for Real-Time Radiance Field Rendering. In *3DV*, pages 1248–1257, 2026. doi: 10.1109/3DV69130.2026.00123. 3, 13

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *SIGGRAPH*, 2024. 13

Yi-Hua Huang, Ming-Xian Lin, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Deformable Radial Kernel Splatting. In *CVPR*, 2025. 13

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4), 2023. 2, 3, 7, 9, 13, 18

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*, 2024. 3, 8, 13, 18

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3D Gaussian Representation for Radiance Field. In *CVPR*, 2024. 3

Haolin Li, Jinyang Liu, Mario Sznaier, and Octavia Camps. 3D-HGS: 3D Half-Gaussian Splatting. In *CVPR*, 2025. 3

Zimu Liao, Jifeng Ding, Siwei Cui, Ruixuan Gong, Boni Hu, Yi Wang, Hengjie Li, Hui Wang, Xingcheng Zhang, and Rong Fu. TC-GS: A Faster Gaussian Splatting Module Utilizing Tensor Cores. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pages 1–9, 2025. 8

Rong Liu, Dylan Sun, Meida Chen, Yue Wang, and Andrew Feng. Deformable Beta Splatting. In *SIGGRAPH*, 2025a. 2, 3

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*, 2026. 1, 2, 3, 6, 7, 13, 16

Xiaofeng Liu, Guanchen Meng, Chongyang Feng, Risheng Liu, Zhongxuan Luo, and Xin Fan. TNT-GS: Truncated and Tailored Gaussian Splatting. In *ACM MM*, 2025b. 3

Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis. In *3DV*, 2024. 3

Alexander Mai, Trevor Hedstrom, George Kopanas, Janne Kontkanen, Falko Kuester, and Jonathan T Barron. Radiance Meshes for Volumetric Reconstruction. In *CVPR*, pages 8267–8277, 2026. 3, 13

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*, 2020. 7, 13, 18

Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3D Scene Representation via Self-Organizing Gaussian Grids. In *ECCV*, 2024. 3

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Transactions on Graphics*, 41(4):102:1–102:15, 2022. doi: 10.1145/3528223.3530127. 13

Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable Neural Radiance Fields. In *ICCV*, 2021a. 3

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. *ACM Transactions on Graphics*, 40(6), 2021b. 3

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *CVPR*, 2021. 3, 7, 18

Haoxuan Qu, Zhuoling Li, Hossein Rahmani, Yujun Cai, and Jun Liu. DisC-GS: Discontinuity-aware Gaussian Splatting. In *NeurIPS*, 2024. 3, 13

Jin-Chuan Shi, Miao Wang, Hao-Bin Duan, and Shao-Hua Guan. Language Embedded 3D Gaussians for Open-Vocabulary Scene Understanding. In *CVPR*, 2024. 3

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*, pages 21750–21760, 2025. 3

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861. 7

Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. Multi-Scale 3D Gaussian Splatting for Anti-Aliased Rendering. In *CVPR*, 2024. 3

Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. In *ICLR*, 2024. 3, 18

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-Splatting: Alias-free 3D Gaussian Splatting. In *CVPR*, 2024. 3, 13

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*, 2018. 7

Ziyu Zhang, Binbin Huang, Hanqing Jiang, Liyang Zhou, Xiaojun Xiang, and Shuhan Shen. Quadratic Gaussian Splatting: High Quality Surface Reconstruction with Second-order Geometric Primitives. In *ICCV*, 2025. 3, 13

Junyu Zhou, Yuyang Huang, Wenrui Dai, Junni Zou, Ziyang Zheng, Nuowen Kan, Chenglin Li, and Hongkai Xiong. 3DGabSplat: 3D Gabor Splatting for Frequency-adaptive Radiance Field Rendering. In *ACM MM*, 2025. 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 *CVPR*, 2024. 3

Jialin Zhu, Jiangbei Yue, Feixiang He, and He Wang. 3D Student Splatting and Scooping. In *CVPR*, 2025. 2, 3

A Comparison with State-of-the-art Methods

The main paper compares each direct method against its covariance-derived counterpart in matched pairs to isolate the contribution of conditioning. Here we situate *d*BS against a broader range of static-scene methods on Mip-NeRF 360 [Barron et al., 2022] and NeRF Synthetic [Mildenhall et al., 2020], including implicit methods (Instant-NGP, Mip-NeRF 360, Zip-NeRF), explicit splatting methods, and recent primitive-based radiance models. Table 4 reports results: *d*BS achieves the best PSNR on Mip-NeRF 360 (28.70 dB), ties for best PSNR on NeRF Synthetic (34.53 dB), and remains competitive on the remaining metrics. Compared to its covariance-derived counterpart UBS [Liu et al., 2026], *d*BS leads on PSNR and SSIM across both benchmarks and on LPIPS for NeRF Synthetic. These results assess competitiveness across metrics.

Evaluation protocol. UBS and *d*BS in Table 4 are re-evaluated under the same 3DGS protocol described in Section 5; UBS numbers therefore differ from those reported in the original UBS paper [Liu et al., 2026].

Table 4: Comparison with SOTA methods on Mip-NeRF360 [Barron et al., 2022] and NeRF Synthetic [Mildenhall et al., 2020]. Literature baselines retain their reported protocols; UBS and *d*BS use the same 3DGS evaluation protocol.

	Methods	Mip-NeRF360			NeRF Synthetic		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
implicit	Instant-NGP [Müller et al., 2022]	25.51	0.684	0.398	33.18	0.959	0.055
	Mip-NeRF360 [Barron et al., 2022]	27.69	0.792	0.237	33.25	0.962	0.039
	Zip-NeRF [Barron et al., 2023]	28.54	0.828	0.189	33.10	0.971	0.031
explicit	3DGS [Kerbl et al., 2023]	27.20	0.815	0.214	33.31	0.969	0.037
	GES [Hamdi et al., 2024]	26.91	0.794	0.250	-	-	-
	2DGS [Huang et al., 2024]	27.04	0.805	0.297	33.07	-	-
	Mip-Splatting [Yu et al., 2024]	27.79	0.827	0.203	33.33	0.969	0.039
	3DGS-MCMC [Kheradmand et al., 2024]	28.29	0.840	0.210	33.80	0.970	0.040
	DRKS [Huang et al., 2025]	26.76	0.787	0.236	33.82	-	-
	Textured GS [Chao et al., 2025]	27.35	0.827	0.186	33.24	0.967	0.043
	Quadratic GS [Zhang et al., 2025]	27.39	0.813	0.213	-	-	-
	DisC-GS [Qu et al., 2024]	28.01	0.833	0.189	-	-	-
	Triangle Splatting [Held et al., 2026]	27.16	0.814	0.191	-	-	-
	Radiance Meshes [Mai et al., 2026]	27.16	0.819	0.255	-	-	-
	Spherical Voronoi [Di Sario et al., 2026]	28.57	0.835	0.230	34.53	0.973	0.032
	UBS[Liu et al., 2026]	28.50	0.841	0.183	34.48	0.975	0.026
	*d*BS (**Ours**)	28.70	0.842	0.185	34.53	0.975	0.025

B Conditional Slicing Algorithms

The fused CUDA kernels for *d*GS and *d*BS are summarized below. Both share the same precision-Cholesky distance computation $\mathbf{L}^\top \boldsymbol{\delta}$ and the same factorized regression structure for position; they differ only in the opacity decay function applied to that distance.

Algorithm 1 *d*GS Conditional Slicing

Require: Primitive parameters: μ_p, μ_q, α, $\mathbf{S}$, $\boldsymbol{\Theta}_{pq}$, $\mathbf{L}$, $\boldsymbol{\Lambda}$
Require: Query point: q (view/time)
$\boldsymbol{\delta} \leftarrow q - \mu_q$
$\alpha_{\text{cond}} \leftarrow \alpha \cdot \exp(-\lambda_o \, \|\mathbf{L}^\top \boldsymbol{\delta}\|^2)$ ▷ Gaussian opacity decay
if *d*GS-O variant **then**
  return $\alpha_{\text{cond}}, \mu_p$ ▷ Early exit for opacity only
end if
$\mathbf{V}_{pq} \leftarrow \text{mean}(\text{diag}(\mathbf{S})) \cdot \boldsymbol{\Theta}_{pq}$
$\tilde{\mathbf{V}}_{qq} \leftarrow \text{diag}(\boldsymbol{\Lambda})\, \mathbf{L}\mathbf{L}^\top$
$\mu_{\text{cond}} \leftarrow \mu_p + \mathbf{V}_{pq} \cdot \tilde{\mathbf{V}}_{qq} \cdot \boldsymbol{\delta}$
return $\alpha_{\text{cond}}, \mu_{\text{cond}}$

Algorithm 2 *d*BS Conditional Slicing

Require: Primitive parameters: μ_p, μ_q, α, $\mathbf{S}$, $\mathbf{\Theta}_{pq}$, $\mathbf{L}$, activated bandwidth $\boldsymbol{\beta}$
Require: Query point: q (view/time)
$\boldsymbol{\delta} \leftarrow q - \mu_q$
$\mathbf{z} \leftarrow \mathbf{L}^\top \boldsymbol{\delta}$
$\alpha_{\text{cond}} \leftarrow \alpha \cdot \prod_{i=1}^{C} (1 - \tanh(z_i^2))^{\beta_i}$ ▷ Beta opacity decay
$\mathbf{\Lambda}_\beta \leftarrow \min(\boldsymbol{\beta}/4,\, \mathbf{1})$ ▷ Beta-derived coupling
$\mathbf{V}_{pq} \leftarrow \text{mean}(\text{diag}(\mathbf{S})) \cdot \mathbf{\Theta}_{pq}$
$\tilde{\mathbf{V}}_{qq} \leftarrow \text{diag}(\mathbf{\Lambda}_\beta)\, \mathbf{L}\mathbf{L}^\top$
$\mu_{\text{cond}} \leftarrow \mu_p + \mathbf{V}_{pq} \cdot \tilde{\mathbf{V}}_{qq} \cdot \boldsymbol{\delta}$
return α_{cond}, μ_{cond}

C Direct Conditional Coordinates for Joint Gaussian Primitives

The operational quantities underlying the direct parameters define coordinates on the space of joint Gaussian covariances. This establishes that the query precision, query-to-position regression operator, and rendered spatial covariance used by *d*GS are compatible with a valid joint Gaussian, even though *d*GS never constructs that covariance. The unrestricted coordinate map is a global diffeomorphism; *d*GS directly optimizes a structured factorization of these rendering-aligned coordinates.

Let the spatial dimension be $p = 3$, let the query dimension be C, and let

$$\mathbf{\Sigma}_{\text{cond}} \in \mathbb{S}^p_{++}, \qquad \mathbf{V}_{qq} \in \mathbb{S}^C_{++}, \qquad \mathbf{M} \in \mathbb{R}^{p \times C}. \tag{10}$$

Here $\mathbf{\Sigma}_{\text{cond}}$ denotes the spatial covariance stored and rendered by *d*GS. In the reconstructed joint-Gaussian interpretation below, it is the conditional spatial covariance and should not be confused with the spatial marginal block $\mathbf{\Sigma}^{\text{joint}}_{pp}$. $\mathbf{V}_{qq}$ is the query-space precision, and $\mathbf{M}$ is the effective query-to-position regression operator. In *d*GS, these operational quantities are

$$\mathbf{V}_{qq} = \mathbf{L}\mathbf{L}^\top, \qquad \mathbf{M} = \mathbf{V}_{pq}\, \text{diag}(\mathbf{\Lambda}) \mathbf{V}_{qq}, \tag{11}$$

while $\mathbf{\Sigma}_{\text{cond}}$ remains query-independent.

Proposition C.1 (Conditional-coordinate parameterization)**.** *Define*

$$\mathbf{\Psi}(\mathbf{\Sigma}_{\text{cond}}, \mathbf{M}, \mathbf{V}_{qq}) = \begin{pmatrix} \mathbf{\Sigma}_{\text{cond}} + \mathbf{M}\mathbf{V}_{qq}^{-1}\mathbf{M}^\top & \mathbf{M}\mathbf{V}_{qq}^{-1} \\ \mathbf{V}_{qq}^{-1}\mathbf{M}^\top & \mathbf{V}_{qq}^{-1} \end{pmatrix}. \tag{12}$$

Then:

(i) $\mathbf{\Psi}(\mathbf{\Sigma}_{\text{cond}}, \mathbf{M}, \mathbf{V}_{qq})$ *is symmetric positive-definite;*

(ii) its query marginal has precision $\mathbf{V}_{qq}$*;*

(iii) conditioning at query $\mathbf{q}$ *gives*

$$\boldsymbol{\mu}_{p|q} = \boldsymbol{\mu}_p + \mathbf{M}(\mathbf{q} - \boldsymbol{\mu}_q), \qquad \mathbf{\Sigma}_{p|q} = \mathbf{\Sigma}_{\text{cond}}; \tag{13}$$

(iv) the map

$$\mathbf{\Psi} : \mathbb{S}^p_{++} \times \mathbb{R}^{p \times C} \times \mathbb{S}^C_{++} \longrightarrow \mathbb{S}^{p+C}_{++} \tag{14}$$

is a smooth bijection.

Proof. The covariance in Equation (12) has the block-triangular factorization

$$\mathbf{\Psi} = \begin{pmatrix} \mathbf{I}_p & \mathbf{M} \\ \mathbf{0} & \mathbf{I}_C \end{pmatrix} \begin{pmatrix} \mathbf{\Sigma}_{\text{cond}} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_{qq}^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{I}_p & \mathbf{0} \\ \mathbf{M}^\top & \mathbf{I}_C \end{pmatrix}. \tag{15}$$

The outer factors are invertible and the middle factor is positive-definite, so $\mathbf{\Psi} \succ \mathbf{0}$. Its blocks satisfy

$$\mathbf{\Sigma}_{qq} = \mathbf{V}_{qq}^{-1}, \qquad \mathbf{\Sigma}_{pq} = \mathbf{M}\mathbf{V}_{qq}^{-1}, \qquad \mathbf{\Sigma}_{pp} = \mathbf{\Sigma}_{\text{cond}} + \mathbf{M}\mathbf{V}_{qq}^{-1}\mathbf{M}^\top. \tag{16}$$

Consequently,

$$\boldsymbol{\Sigma}_{pq}\boldsymbol{\Sigma}_{qq}^{-1} = \mathbf{M}\mathbf{V}_{qq}^{-1}\mathbf{V}_{qq} = \mathbf{M}, \tag{17}$$

which gives the conditional mean in Equation (13). The conditional covariance is

$$\boldsymbol{\Sigma}_{p|q} = \boldsymbol{\Sigma}_{pp} - \boldsymbol{\Sigma}_{pq}\boldsymbol{\Sigma}_{qq}^{-1}\boldsymbol{\Sigma}_{pq}^{\top} \tag{18}$$

$$= \boldsymbol{\Sigma}_{\text{cond}} + \mathbf{M}\mathbf{V}_{qq}^{-1}\mathbf{M}^{\top} - \mathbf{M}\mathbf{V}_{qq}^{-1}\mathbf{V}_{qq}\mathbf{V}_{qq}^{-1}\mathbf{M}^{\top} = \boldsymbol{\Sigma}_{\text{cond}}. \tag{19}$$

Conversely, let

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{pp} & \boldsymbol{\Sigma}_{pq} \\ \boldsymbol{\Sigma}_{pq}^{\top} & \boldsymbol{\Sigma}_{qq} \end{pmatrix} \succ \mathbf{0}. \tag{20}$$

Its conditional coordinates are uniquely recovered by

$$\mathbf{V}_{qq} = \boldsymbol{\Sigma}_{qq}^{-1}, \qquad \mathbf{M} = \boldsymbol{\Sigma}_{pq}\boldsymbol{\Sigma}_{qq}^{-1}, \qquad \boldsymbol{\Sigma}_{\text{cond}} = \boldsymbol{\Sigma}_{pp} - \boldsymbol{\Sigma}_{pq}\boldsymbol{\Sigma}_{qq}^{-1}\boldsymbol{\Sigma}_{pq}^{\top}. \tag{21}$$

The Schur complement theorem gives $\boldsymbol{\Sigma}_{\text{cond}} \succ \mathbf{0}$. Both the forward map and this inverse are smooth on their positive-definite domains, proving the claim. □

Parameter count. The dimensions match exactly:

$$\frac{p(p+1)}{2} + pC + \frac{C(C+1)}{2} = \frac{(p+C)(p+C+1)}{2}. \tag{22}$$

Thus unrestricted $(\boldsymbol{\Sigma}_{\text{cond}}, \mathbf{M}, \mathbf{V}_{qq})$ are a complete, nonredundant coordinate system for joint Gaussian covariances. The actual *d*GS factorization in Equation (11) additionally constrains $\mathbf{V}_{pq}$ by spatial scaling and normalization and constrains $\boldsymbol{\Lambda} \in [0,1]^C$. It therefore selects a structured subfamily of this chart; we do not claim that the internal $(\mathbf{V}_{pq}, \boldsymbol{\Lambda})$ factorization is itself bijective.

Corollary C.2 (Exact implicit joint Gaussian for *d*GS)**.** *Define the opacity precision and effective regression operator as*

$$\boldsymbol{\Pi} = 2\lambda_o\mathbf{V}_{qq}, \qquad \mathbf{M} = \mathbf{V}_{pq}\,\mathrm{diag}(\boldsymbol{\Lambda})\mathbf{V}_{qq}. \tag{23}$$

Then the implemented dGS opacity exponent and position update are, respectively, the query-marginal quadratic form and conditional-mean regression of the valid joint covariance

$$\boxed{\boldsymbol{\Sigma}_{\text{dGS}}^{\star} = \boldsymbol{\Psi}(\boldsymbol{\Sigma}_{\text{cond}}, \mathbf{M}, \boldsymbol{\Pi}) = \begin{pmatrix} \boldsymbol{\Sigma}_{\text{cond}} + \mathbf{M}\boldsymbol{\Pi}^{-1}\mathbf{M}^{\top} & \mathbf{M}\boldsymbol{\Pi}^{-1} \\ \boldsymbol{\Pi}^{-1}\mathbf{M}^{\top} & \boldsymbol{\Pi}^{-1} \end{pmatrix}.} \tag{24}$$

Equivalently, its blocks are

$$\boldsymbol{\Sigma}_{qq}^{\star} = \frac{1}{2\lambda_o}\mathbf{V}_{qq}^{-1}, \qquad \boldsymbol{\Sigma}_{pq}^{\star} = \frac{1}{2\lambda_o}\mathbf{V}_{pq}\,\mathrm{diag}(\boldsymbol{\Lambda}), \tag{25}$$

$$\boldsymbol{\Sigma}_{pp}^{\star} = \boldsymbol{\Sigma}_{\text{cond}} + \frac{1}{2\lambda_o}\mathbf{V}_{pq}\,\mathrm{diag}(\boldsymbol{\Lambda})\mathbf{V}_{qq}\,\mathrm{diag}(\boldsymbol{\Lambda})\mathbf{V}_{pq}^{\top}. \tag{26}$$

The resulting conditional mean and covariance are exactly

$$\boldsymbol{\mu}_{\text{cond}} = \boldsymbol{\mu}_p + \mathbf{V}_{pq}\,\mathrm{diag}(\boldsymbol{\Lambda})\mathbf{V}_{qq}(\mathbf{q} - \boldsymbol{\mu}_q), \qquad \boldsymbol{\Sigma}_{p|q} = \boldsymbol{\Sigma}_{\text{cond}}. \tag{27}$$

Proof. Apply Proposition C.1 with query precision $\boldsymbol{\Pi}$ and regression operator $\mathbf{M}$. The query marginal has precision $(\boldsymbol{\Sigma}_{qq}^{\star})^{-1} = \boldsymbol{\Pi} = 2\lambda_o\mathbf{V}_{qq}$, so its conventional Gaussian exponent is

$$-\frac{1}{2}\boldsymbol{\delta}^{\top}(\boldsymbol{\Sigma}_{qq}^{\star})^{-1}\boldsymbol{\delta} = -\lambda_o\boldsymbol{\delta}^{\top}\mathbf{V}_{qq}\boldsymbol{\delta}, \tag{28}$$

which is exactly the implemented opacity exponent. Moreover,

$$\boldsymbol{\Sigma}_{pq}^{\star}(\boldsymbol{\Sigma}_{qq}^{\star})^{-1} = \frac{1}{2\lambda_o}\mathbf{V}_{pq}\,\mathrm{diag}(\boldsymbol{\Lambda})(2\lambda_o\mathbf{V}_{qq}) = \mathbf{M}, \tag{29}$$

which is the implemented conditional-mean regression. The conditional covariance follows from Proposition C.1, and direct substitution gives the blocks in Equations (25) and (26). □

As in standard Gaussian splatting, opacity uses an unnormalized Gaussian kernel rather than a normalized probability density. Consequently, the probabilistic interpretation concerns the quadratic form, conditional mean, and covariance structure; it does not require including the Gaussian density normalization factor in the renderer.

Corollary C.3 (Decoupling and full coupling)**.** *When $\boldsymbol{\Lambda} = \mathbf{0}$, Equation (24) is block diagonal and position is query-independent although opacity remains query-dependent. When $\boldsymbol{\Lambda} = \mathbf{1}$, the full regression coupling is active, $\mathbf{M} = \mathbf{V}_{pq}\mathbf{V}_{qq}$. Intermediate values continuously weight the precision-transformed query channels.*

C.1 Determinant Structure

The outer factors in Equation (15) are unit block-triangular. Therefore

$$\boxed{\det \mathbf{\Psi} = \det(\mathbf{\Sigma}_{\text{cond}}) \det(\mathbf{V}_{qq}^{-1}) = \frac{\det \mathbf{\Sigma}_{\text{cond}}}{\det \mathbf{V}_{qq}}.} \tag{30}$$

The regression operator $\mathbf{M}$ does not appear. Geometrically, $\mathbf{M}$ acts as a block shear: it changes cross-covariance and the conditional mean without changing total covariance volume. The two determinant factors separately measure spatial conditional volume and query-space volume.

C.2 Volume-Element Jacobian of the Coordinate Transformation

Use the standard Lebesgue coordinates on symmetric and rectangular matrices. The conditional-coordinate map has a closed-form volume-element Jacobian.

Proposition C.4 (Coordinate-volume factor)**.** *For the map in Equation* (12)*,*

$$\boxed{|\det D\mathbf{\Psi}| = (\det \mathbf{V}_{qq})^{-(p+C+1)}.} \tag{31}$$

For spatial dimension $p = 3$, hence

$$|\det D\mathbf{\Psi}| = (\det \mathbf{V}_{qq})^{-(C+4)}. \tag{32}$$

Proof. Factor the transformation into three block-triangular steps. First set $\mathbf{Q} = \mathbf{V}_{qq}^{-1}$. Inversion on $\mathbb{S}_{++}^{C}$ has Jacobian

$$\left|\det \frac{\partial \mathbf{Q}}{\partial \mathbf{V}_{qq}}\right| = (\det \mathbf{Q})^{C+1} = (\det \mathbf{V}_{qq})^{-(C+1)}. \tag{33}$$

Second, $\mathbf{\Sigma}_{pq} = \mathbf{M}\mathbf{Q}$. Right multiplication of a $p \times C$ matrix by $\mathbf{Q}$ contributes

$$(\det \mathbf{Q})^{p} = (\det \mathbf{V}_{qq})^{-p}. \tag{34}$$

Finally, $\mathbf{\Sigma}_{pp} = \mathbf{\Sigma}_{\text{cond}} + \mathbf{M}\mathbf{Q}\mathbf{M}^{\top}$ is a translation in the $\mathbf{\Sigma}_{\text{cond}}$ coordinates and contributes determinant one. Cross derivatives between these steps lie below the corresponding diagonal Jacobian blocks and do not change the determinant. Multiplying the diagonal factors gives Equation (31). □

For every $\mathbf{\Sigma}_{\text{cond}} \succ \mathbf{0}$ and $\mathbf{V}_{qq} \succ \mathbf{0}$, the coordinate-volume factor is finite and nonzero. Thus the conditional-coordinate map has no singularities in the interior of its SPD domain. The domain retains the usual boundary behavior when either SPD factor approaches its boundary, including when the conditional covariance or corresponding query covariance approaches singularity. The construction $\mathbf{V}_{qq} = \mathbf{L}\mathbf{L}^{\top}$ with a strictly positive Cholesky diagonal keeps *d*GS in the interior with respect to query precision. This statement concerns the ambient $(\mathbf{\Sigma}_{\text{cond}}, \mathbf{M}, \mathbf{V}_{qq})$ chart, not uniqueness of *d*GS's structured internal factorization of $\mathbf{M}$.

Interpretation. The rendered spatial covariance, query precision, and regression operator are Gaussian regression coordinates. Direct conditioning optimizes these operational quantities rather than repeatedly recovering them from an ambient covariance by inversion and Schur complementation. The coupling is a block shear with an explicit inverse and determinant. Within the structured dGS factorization, $\mathbf{V}_{pq}$ supplies the spatial-scale prior, while $\operatorname{diag}(\mathbf{\Lambda})\mathbf{V}_{qq}$ completes the regression operator and exposes learnable coupling.

D Covariance-Derived versus Direct Regression Coordinates

The conditional mean depends on the regression operator $\mathbf{\Sigma}_{pq}\mathbf{\Sigma}_{qq}^{-1}$, not on the cross-covariance $\mathbf{\Sigma}_{pq}$ alone. This section derives that operator for N-DGS [Gao et al., 2025a,b] and contrasts its implicit covariance coordinates with the direct conditional coordinates of *d*GS. UBS [Liu et al., 2026] inherits the same covariance-derived structure.

Block-Cholesky derivation. Partition the joint Cholesky factor into spatial and query blocks:

$$\mathcal{L} = \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{B} & \mathbf{D} \end{pmatrix}, \qquad \mathbf{A} \in \mathbb{R}^{p\times p}, \quad \mathbf{B} \in \mathbb{R}^{C\times p}, \quad \mathbf{D} \in \mathbb{R}^{C\times C}. \tag{35}$$

Here $\mathbf{A}$ and $\mathbf{D}$ are lower triangular. Multiplication gives

$$\boldsymbol{\Sigma} = \mathcal{L}\mathcal{L}^\top = \begin{pmatrix} \mathbf{A}\mathbf{A}^\top & \mathbf{A}\mathbf{B}^\top \\ \mathbf{B}\mathbf{A}^\top & \mathbf{B}\mathbf{B}^\top + \mathbf{D}\mathbf{D}^\top \end{pmatrix}, \tag{36}$$

and therefore

$$\boldsymbol{\Sigma}_{pp} = \mathbf{A}\mathbf{A}^\top, \qquad \boldsymbol{\Sigma}_{pq} = \mathbf{A}\mathbf{B}^\top, \qquad \boldsymbol{\Sigma}_{qq} = \mathbf{B}\mathbf{B}^\top + \mathbf{D}\mathbf{D}^\top. \tag{37}$$

The actual N-DGS query-to-position regression operator is

$$\boxed{\mathbf{M}_{\text{N-DGS}} = \mathbf{A}\mathbf{B}^\top \left(\mathbf{B}\mathbf{B}^\top + \mathbf{D}\mathbf{D}^\top\right)^{-1}.} \tag{38}$$

Thus $\mathbf{A}$ affects both spatial covariance and cross-covariance; $\mathbf{B}$ affects both cross-covariance and query covariance; and $\mathbf{D}$ also affects query covariance. The operational regression is available only after forming and inverting the query block.

A valid cross-covariance bound. For spatial row i,

$$(\boldsymbol{\Sigma}_{pq})_{i,:} = \mathbf{A}_{i,:}\mathbf{B}^\top, \qquad \|(\boldsymbol{\Sigma}_{pq})_{i,:}\|_2 \le \|\mathbf{A}_{i,:}\|_2 \, \|\mathbf{B}\|_2. \tag{39}$$

Moreover,

$$\|\mathbf{A}_{i,:}\|_2^2 = (\mathbf{A}\mathbf{A}^\top)_{ii} = (\boldsymbol{\Sigma}_{pp})_{ii}. \tag{40}$$

In N-DGS, bounded Cholesky coefficients are multiplied by the query diagonal scales $\mathbf{s}_q$. Thus $\mathbf{B} = \operatorname{diag}(\mathbf{s}_q)\widetilde{\mathbf{B}}$ with $|\widetilde{B}_{jk}| \le 1$, giving

$$\|\mathbf{B}\|_2 \le \|\mathbf{B}\|_F \le \sqrt{p}\,\|\mathbf{s}_q\|_2, \tag{41}$$

so

$$\boxed{\|(\boldsymbol{\Sigma}_{pq})_{i,:}\|_2 \le \sqrt{p}\,\|\mathbf{s}_q\|_2 \sqrt{(\boldsymbol{\Sigma}_{pp})_{ii}}.} \tag{42}$$

The off-diagonal entries of $\mathbf{A}$ contribute both to $\boldsymbol{\Sigma}_{pq}$ and to the actual marginal spatial extent $(\boldsymbol{\Sigma}_{pp})_{ii}$. They should therefore not be interpreted as producing cross-covariance independently of spatial scale.

Cross-covariance is not displacement. The rendered N-DGS shift is

$$\Delta\boldsymbol{\mu}_{\text{N-DGS}} = \boldsymbol{\Sigma}_{pq}\boldsymbol{\Sigma}_{qq}^{-1}\boldsymbol{\delta} = \mathbf{M}_{\text{N-DGS}}\boldsymbol{\delta}. \tag{43}$$

A bound on $\boldsymbol{\Sigma}_{pq}$ alone therefore does not bound the shift: the inverse query covariance and the query deviation also enter. In Euclidean coordinates, a narrow or ill-conditioned query covariance can amplify the regression operator.

Invariant Gaussian bound. A basis-independent statement follows from the normalized cross-covariance

$$\mathbf{K} = \boldsymbol{\Sigma}_{pp}^{-1/2}\boldsymbol{\Sigma}_{pq}\boldsymbol{\Sigma}_{qq}^{-1/2}. \tag{44}$$

Positive-definiteness of the joint covariance implies $\|\mathbf{K}\|_2 < 1$ by the Schur complement. With the whitened query deviation $\mathbf{h} = \boldsymbol{\Sigma}_{qq}^{-1/2}\boldsymbol{\delta}$,

$$\Delta\boldsymbol{\mu}_{\text{N-DGS}} = \boldsymbol{\Sigma}_{pp}^{1/2}\mathbf{K}\mathbf{h}, \tag{45}$$

and hence

$$\boxed{\left\|\boldsymbol{\Sigma}_{pp}^{-1/2}\Delta\boldsymbol{\mu}_{\text{N-DGS}}\right\|_2 \le \left\|\boldsymbol{\Sigma}_{qq}^{-1/2}\boldsymbol{\delta}\right\|_2.} \tag{46}$$

Measured in marginal spatial standard-deviation units, the shift is bounded by the query deviation in query standard-deviation units, irrespective of basis or Cholesky ordering.

Direct regression coordinates in *d*GS. The direct formulation parameterizes

$$\mathbf{V}_{qq} = \mathbf{L}\mathbf{L}^\top, \qquad \mathbf{M}_{\mathrm{dGS}} = \mathbf{V}_{pq}\,\mathrm{diag}(\mathbf{\Lambda})\mathbf{V}_{qq}, \qquad \mathbf{\Sigma}_{\mathrm{cond}} \text{ independently.} \tag{47}$$

Its spatial coupling factor is normalized as

$$\mathbf{V}_{pq} = \bar{s}\mathbf{\Theta}_{pq}, \qquad \|\mathbf{\Theta}_{pq}\|_F = 1, \tag{48}$$

which gives

$$\|\mathbf{V}_{pq}\|_F = \bar{s}, \qquad \|\mathbf{V}_{pq}^{(i)}\|_2 \le \bar{s}. \tag{49}$$

This is an explicit spatial-scale prior on the coupling factor, not a bound on the effective regression or rendered shift. Those additionally depend on $\mathrm{diag}(\mathbf{\Lambda})\mathbf{V}_{qq}$ and $\boldsymbol{\delta}$.

The conditional-coordinate theorem in Appendix C makes the comparison precise. N-DGS indirectly parameterizes conditional covariance, regression, and query precision through the ambient joint Cholesky factor. *d*GS makes the corresponding operational quantities explicit and exposes a learnable coupling strength, without forming a query-block inverse during slicing. The benefit is this computational and parameterization separation, rather than an invalid geometric bound for N-DGS.

Empirically, removing the spatial factor $\bar{s}$ degrades PSNR by 2.65 dB on NeRF Synthetic and 2.97 dB on 6DGS-PBR (Table 3), supporting the spatial-scale prior on $\mathbf{V}_{pq}$. On two trained MCMC scenes with six views each, opacity-weighted $\|\Delta\boldsymbol{\mu}\|/\bar{s}$ has median 0.29–0.30 and 99th percentile 4.8–7.0 for N-DGS, versus 2.9–4.8 at the 99th percentile for *d*GS. The ordering reverses when normalized by the smallest principal axis, supporting $\bar{s}$ as an isotropic prior rather than a per-axis optimum.

E Results with Standard Densification

The main paper reports MCMC densification [Kheradmand et al., 2024], the protocol used by UBS originally. Here we additionally evaluate the Gaussian-kernel pair (*d*GS vs. N-DGS) under standard 3DGS densification [Kerbl et al., 2023], the protocol used by N-DGS originally. For dynamic scenes, these standard-densification runs use batch size 4, following the original N-DGS protocol. Table 5 summarizes quality (PSNR, SSIM, LPIPS) and efficiency metrics across all five datasets: three static (NeRF Synthetic [Mildenhall et al., 2020], Mip-NeRF 360 [Barron et al., 2022], 6DGS-PBR [Gao et al., 2025a]) and two dynamic (D-NeRF [Pumarola et al., 2021], 7DGS-PBR [Gao et al., 2025b]).

Table 5: Standard 3DGS densification: scene means from one run per scene (#Gauss in K, training time in minutes; dynamic batch size 4). $\mathrm{FPS}_{\mathrm{CUDA}}$ is an arithmetic scene mean. For 3DGS [Kerbl et al., 2023] and 4DGS [Yang et al., 2024] comparisons, see the original N-DGS [Gao et al., 2025a,b].

	Dataset	Method	#Gauss	Time ↓	$\mathrm{FPS}_{\mathrm{CUDA}}$ ↑	PSNR ↑	SSIM ↑	LPIPS ↓
STATIC	NeRF Synthetic	N-DGS	**171**	6.35	615.57	33.38	0.970	0.032
		*d*GS-O (**Ours**)	201	**5.83**	**754.98**	33.84	0.971	0.030
		*d*GS (**Ours**)	201	6.84	704.84	**33.91**	**0.972**	**0.029**
	Mip-NeRF 360	N-DGS	**1273**	27.09	147.46	26.40	0.739	0.316
		*d*GS-O (**Ours**)	2363	**25.58**	**279.30**	27.18	0.779	0.252
		*d*GS (**Ours**)	2463	28.21	237.82	**27.37**	**0.781**	**0.251**
	6DGS-PBR	N-DGS	**84**	**26.77**	678.40	37.11	0.971	0.054
		*d*GS-O (**Ours**)	118	30.72	**696.31**	36.54	0.965	0.059
		*d*GS (**Ours**)	134	33.80	653.44	**37.75**	**0.972**	**0.052**
DYNAMIC	D-NeRF	N-DGS	**69**	14.08	852.01	33.13	0.968	0.038
		*d*GS-O (**Ours**)	70	**13.20**	**948.77**	30.92	0.959	0.044
		*d*GS (**Ours**)	85	15.56	818.41	**33.59**	**0.971**	**0.032**
	7DGS-PBR	N-DGS	**162**	38.14	709.83	31.75	0.955	0.055
		*d*GS-O (**Ours**)	227	**34.58**	**835.24**	31.70	0.955	0.053
		*d*GS (**Ours**)	226	36.96	753.77	**32.18**	**0.956**	**0.050**

Static Scenes. On NeRF Synthetic, both variants improve quality: *d*GS-O by +0.46 dB and *d*GS by +0.53 dB, with rendering speedups of 23% and 15% respectively. On Mip-NeRF 360, efficiency gains are most pronounced: *d*GS-O achieves 89% faster rendering with +0.78 dB improvement, while *d*GS achieves 61% faster rendering with +0.97 dB improvement. Notably, *d*GS-O uses 86% more primitives (2363K vs 1273K) yet remains faster, demonstrating that per-primitive computation cost

dominates over primitive count. On 6DGS-PBR, *d*GS-O shows modest 3% speedup but -0.57 dB quality drop, while *d*GS recovers and exceeds baseline quality (+0.64 dB).

Dynamic Scenes. Position conditioning proves essential for temporal dynamics. On D-NeRF, *d*GS-O suffers -2.21 dB quality drop despite 11% faster rendering, while *d*GS with learnable coupling via Λ exceeds N-DGS by +0.46 dB, demonstrating adaptive coupling surpasses fixed coupling. On 7DGS-PBR combining dynamics with angular lighting, *d*GS-O achieves 18% faster rendering with minimal quality loss (-0.05 dB), while *d*GS improves quality by +0.43 dB with 6% faster rendering.

F UBS Formulation

The released UBS implementation used in our experiments evaluates $\beta_i = 4\exp(\beta_i^{\text{raw}})$ and $\mathbf{\Lambda}_\beta = \min(\boldsymbol{\beta}/4, \mathbf{1})$. Its covariance-derived slicing is

$$\mathbf{z}^{\text{UBS}} = \hat{\mathbf{L}}_q^{-1}\boldsymbol{\delta}, \quad d_i = \tanh\big((z_i^{\text{UBS}})^2\big), \tag{50}$$

$$\alpha_{\text{cond}} = \alpha \prod_{i=1}^{C} (1 - d_i)^{\beta_i}, \tag{51}$$

$$\boldsymbol{\mu}_{\text{cond}} = \boldsymbol{\mu}_p + \mathbf{\Sigma}_{pq}\mathbf{\Sigma}_{qq}^{-1}\,\text{diag}(\mathbf{\Lambda}_\beta)\boldsymbol{\delta}, \tag{52}$$

$$\mathbf{\Sigma}_{\text{cond}} = \mathbf{\Sigma}_{pp} - \mathbf{\Sigma}_{pq}\mathbf{\Sigma}_{qq}^{-1}\,\text{diag}(\mathbf{\Lambda}_\beta)\mathbf{\Sigma}_{qp}, \tag{53}$$

where $\hat{\mathbf{L}}_q\hat{\mathbf{L}}_q^\top = \mathbf{\Sigma}_{qq}$. The `tanh` values are capped below one for finite opacity evaluation. *d*BS (Equations (8) and (9)) retains the same opacity decay, bandwidth activation, and clipped coupling. It replaces the triangular solve with $\mathbf{L}^\top\boldsymbol{\delta}$, replaces the covariance-derived regression with $\mathbf{V}_{pq}\,\text{diag}(\mathbf{\Lambda}_\beta)\mathbf{V}_{qq}$, and directly stores $\mathbf{\Sigma}_{\text{cond}}$.

G Parameterization and Evaluation Controls

Unrestricted direct regression. To distinguish direct readout from its structured parameterization, we replace $\mathbf{V}_{pq}\,\text{diag}(\mathbf{\Lambda})\mathbf{V}_{qq}$ with a free matrix $\mathbf{M} \in \mathbb{R}^{3\times C}$, retaining the directly stored conditional covariance and full query precision. This Direct-Unrestricted (DU) variant removes spatial normalization and bounded coupling. Its matrix learning rate is 5×10^{-6}, scaled by the median primitive size to account for the different parameter units and then fixed across scenes. Table 7 shows that unrestricted direct coordinates alone do not explain the quality gains: the structured parameterization performs better on both datasets. This comparison evaluates the structure and optimization coordinates collectively.

Variation across training seeds. Table 8 summarizes three matched seeds for the smallest dBS/UBS margin and for the coupling ablation. The dBS/UBS difference remains quality parity, while learned coupling ranks above both fixed endpoints in all three runs. These repeated-seed results concern NeRF Synthetic; the other benchmark comparisons retain the single-run protocol.

Batch size. On 7DGS-PBR, using batch size four for both Beta methods gives 32.36 dB for UBS and 33.63 dB for *d*BS, a 1.27 dB improvement on the same joint-RGB metric as the main table. The ordering holds on all six scenes, consistent with the 1.26 dB gain at batch size one.

Table 6: Detailed quantitative comparison across all datasets at 30K iterations. N-DGS, dGS-O, and dGS are compared on static benchmarks (NeRF Synthetic, Mip-NeRF 360, 6DGS-PBR) and dynamic benchmarks (7DGS-PBR, D-NeRF).

	Dataset	Scene	N-DGS			*d*GS-O (**Ours**)			*d*GS (**Ours**)		
			PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
STATIC SCENES	NeRF Synthetic	chair	35.75	0.986	0.013	**36.36**	**0.988**	**0.012**	36.26	**0.988**	**0.012**
		drums	26.75	0.956	0.036	26.80	0.957	0.035	**26.92**	**0.958**	**0.034**
		ficus	33.54	0.984	0.014	**35.30**	**0.988**	**0.011**	35.03	0.988	**0.011**
		hotdog	38.10	0.985	0.022	**38.13**	**0.986**	**0.020**	**38.15**	**0.986**	**0.020**
		lego	35.63	0.981	0.019	**35.87**	**0.982**	**0.017**	35.79	**0.982**	**0.017**
		materials	30.78	0.967	0.033	31.00	0.967	0.030	**31.76**	**0.972**	**0.027**
		mic	35.63	0.992	0.006	36.12	**0.993**	**0.005**	**36.25**	**0.993**	**0.005**
		ship	30.85	0.906	0.112	**31.16**	**0.908**	0.107	31.13	**0.908**	**0.106**
		avg	33.38	0.970	0.032	33.84	0.971	0.030	**33.91**	**0.972**	**0.029**
	Mip-NeRF 360	bicycle	22.33	0.554	0.447	23.49	0.644	0.327	**23.51**	**0.648**	**0.318**
		bonsai	33.20	0.940	0.208	33.22	0.949	0.180	**33.46**	**0.951**	**0.175**
		counter	29.26	0.895	0.236	29.73	0.917	0.185	**29.93**	**0.920**	**0.181**
		flowers	18.90	0.409	0.507	19.61	**0.453**	**0.443**	**19.71**	0.452	0.455
		garden	26.19	0.809	0.205	26.77	0.842	0.145	**27.13**	**0.849**	**0.137**
		kitchen	31.17	0.922	0.137	31.72	0.930	0.121	**32.04**	**0.933**	**0.116**
		room	31.60	0.910	0.243	31.83	0.927	0.199	**32.25**	**0.929**	**0.195**
		stump	23.32	0.638	0.413	**26.01**	0.750	**0.275**	26.00	**0.742**	0.295
		treehill	21.66	0.571	0.448	22.21	0.603	0.393	**22.31**	**0.606**	**0.389**
		avg	26.40	0.739	0.316	27.18	0.779	0.252	**27.37**	**0.781**	**0.251**
	6DGS-PBR	bunny	40.87	0.993	0.045	43.44	0.993	0.042	**44.14**	**0.993**	**0.040**
		cloud	41.93	0.991	0.052	43.47	0.992	0.048	**43.53**	**0.992**	**0.045**
		dragon	35.69	0.947	0.039	34.42	0.914	0.051	**36.68**	**0.947**	**0.039**
		explosion	**40.97**	**0.989**	**0.037**	37.81	0.983	0.041	38.95	0.985	0.039
		ct-scan	33.48	0.964	0.057	31.81	0.956	0.067	**34.62**	**0.970**	**0.048**
		smoke	**40.02**	**0.990**	**0.041**	38.20	0.989	0.049	39.28	**0.990**	0.047
		suzanne	26.81	0.929	**0.106**	26.65	0.927	0.112	**27.03**	**0.930**	0.107
		avg	37.11	0.971	0.054	36.54	0.965	0.059	**37.75**	**0.972**	**0.052**
DYNAMIC SCENES	7DGS-PBR	cloud	29.28	0.955	0.075	29.34	0.955	0.075	**29.82**	**0.955**	**0.074**
		dust	36.85	0.955	0.038	36.87	0.955	0.037	**38.39**	**0.958**	**0.031**
		flame	**31.64**	**0.937**	**0.062**	30.38	0.933	0.066	31.52	0.936	0.060
		heart1	34.66	0.983	0.023	35.14	**0.985**	0.021	**35.25**	**0.986**	**0.018**
		heart2	30.99	0.959	0.057	**31.29**	**0.962**	0.052	31.01	**0.962**	**0.049**
		suzanne	27.09	**0.941**	0.072	**27.20**	0.940	**0.069**	27.07	0.940	0.070
		avg	31.75	0.955	0.055	31.70	0.955	0.053	**32.18**	**0.956**	**0.050**
	D-NeRF	b.balls	**34.00**	**0.981**	0.025	32.52	0.978	0.028	33.59	**0.981**	**0.023**
		h.warrior	32.76	0.932	0.093	33.54	0.941	0.085	**34.64**	**0.947**	**0.073**
		hook	31.15	0.960	0.044	28.55	0.937	0.062	**32.94**	**0.970**	**0.030**
		j.jacks	**31.22**	**0.967**	0.043	28.62	0.952	0.054	30.86	0.965	**0.039**
		lego	**28.33**	**0.943**	0.055	27.13	0.939	0.055	28.06	0.942	**0.054**
		mutant	39.38	0.993	0.007	34.55	0.981	0.019	**39.84**	**0.993**	**0.006**
		standup	38.50	0.988	0.013	33.02	0.972	0.029	**38.76**	**0.990**	**0.009**
		trex	29.68	0.978	0.023	29.42	0.975	0.022	**30.08**	**0.978**	**0.021**
		avg	33.13	0.968	0.038	30.92	0.959	0.044	**33.59**	**0.971**	**0.032**

Table 7: Direct-regression control under standard densification. Dataset-mean PSNR (dB), one training run per scene.

Dataset	N-DGS	DU	*d*GS
NeRF Synthetic	33.38	32.91	**33.91**
6DGS-PBR	37.11	36.44	**37.75**

Table 8: NeRF Synthetic PSNR (dB): mean and sample standard deviation of dataset means across three training seeds.

Protocol	Method	PSNR
MCMC	UBS	34.50 ± 0.02
	*d*BS	34.53 ± 0.02
Standard	$\mathbf{\Lambda} = \mathbf{0}$	33.83 ± 0.02
	$\mathbf{\Lambda} = \mathbf{1}$	33.78 ± 0.01
	Learned $\mathbf{\Lambda}$	33.92 ± 0.02

H Conditioning and End-to-End Efficiency

Inference controls. Table 9 compares fused CUDA kernels with inputs already materialized. Cached N-DGS stores query precision, regression, and conditional covariance; the factorized-query control uses $\mathbf{\Sigma}_{qq} = \mathbf{R}\mathbf{S}^2\mathbf{R}^\top$ to obtain the inverse analytically, while still deriving regression and conditional covariance. Direct conditioning is 2.7–2.8× faster than on-the-fly N-DGS, 2.0–2.3× faster than its cached form, and 1.8× faster than the factorized-query control. These timings exclude production input preparation and therefore differ from the production-slicing timings in Table 1. Persistent caching applies to inference; training updates require refreshing the cached quantities.

Table 9: Slicing time (ms) for one million primitives on an A100. Medians of 100 trials after 20 warmups; inputs are materialized before timing.

Parameterization	$C = 3$	$C = 4$
N-DGS, on-the-fly	0.224	0.269
N-DGS, cached	0.178	0.197
N-DGS, factorized query	0.141	0.180
*d*GS, direct	**0.079**	**0.100**

Frame-time decomposition. Slicing is a vertex-like preprocessing stage before projection, tile binning, sorting, and rasterization. If its baseline fraction is f and its speedup is s, an unchanged remainder would give the Amdahl speedup $[(1 - f) + f/s]^{-1}$. In Table 10, slicing accounts for 42–64% of baseline frame time. The downstream code is shared, but the learned primitives have different projected footprints and tile workloads: the measured remainder is 1.10–1.68× slower for *d*GS. The observed end-to-end gains are consequently 1.25–1.91×. These per-scene frame times differ from arithmetic-mean FPS across all nine Mip-NeRF 360 scenes (Table 2).

Table 10: Frame-time decomposition (ms) on trained checkpoints using matched training caps. Each pair lists N-DGS / *d*GS; the remainder includes all post-slicing stages.

Scene	Slicing	Remainder	Total	Speedup
`lego`	1.56 / 0.22	2.19 / 2.42	3.75 / 2.65	1.42×
`counter`	7.38 / 0.57	8.77 / 12.18	16.15 / 12.75	1.27×
`garden`	24.83 / 1.69	13.96 / 18.61	38.79 / 20.30	1.91×
`bicycle`	29.78 / 2.02	24.71 / 41.59	54.49 / 43.61	1.25×

I Limitations and Societal Impact

Limitations. The evaluation covers Gaussian and Beta kernels. Larger-scale scenes may change memory, visibility, and partitioning bottlenecks; other kernels and domains require validation.

Societal impact. Faster neural rendering can benefit simulation, telepresence, robotics, and content creation, but can also facilitate unauthorized reconstruction of private spaces. This work introduces no new data sources or user-facing decision systems.

Existing assets. All datasets and baseline methods used in this work are public research assets and are cited in the paper. We use NeRF Synthetic, Mip-NeRF 360, D-NeRF, 6DGS-PBR, and 7DGS-PBR only for academic evaluation and do not redistribute the datasets or baseline code.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction state the method, matched-pair scope, speedups, and quality claims. The experiments and appendix report the corresponding benchmark results, while limitations and scope are discussed in the conclusion and supplemental limitations section.

2. **Limitations**

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

   Answer: [Yes]

   Justification: The conclusion includes future work, and the supplemental Section I discusses evaluation scope, lack of city-scale/outdoor driving-scale validation, and the need to validate additional kernels and domains.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: Appendices C and D state assumptions and prove the conditional-coordinate correspondence and Gaussian displacement bound.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The experimental setup specifies datasets, metrics, evaluation protocol, optimizer, iteration count, densification protocol, batch-size differences for dynamic MCMC results, initialization, hardware, and CUDA/rasterizer implementation details.

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

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

   Answer: [No]

   Justification: The experiments use cited public benchmark datasets, and the paper provides implementation details. Public code and complete reproduction scripts are not provided.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The experimental setup describes the datasets, evaluation protocol, metrics, training pipeline, optimizer, iteration count, densification settings, batch size for dynamic MCMC runs, and parameter initialization. Additional standard-densification details and per-scene results are provided in the appendix.

7. **Experiment statistical significance**

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

   Answer: [Yes]

Justification: Appendix G reports means and sample standard deviations across three matched seeds for the NeRF Synthetic dBS/UBS comparison and coupling ablation. Other benchmark results are single-run scene averages; the repeated runs do not establish variability across all five datasets.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper states that experiments use a single NVIDIA A100 GPU and reports training time, Gaussian count, and rendering speeds in the main and appendix tables. Kernel benchmarks also report slicing-only forward-pass timing on 1M Gaussians.

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The work uses public benchmark datasets and does not involve human subjects, private data collection, or deployment in user-facing decision systems. The paper also discusses societal impact in the supplement.

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Supplemental Section I discusses beneficial uses such as simulation, telepresence, robotics, and content creation, as well as potential misuse such as unauthorized reconstruction of private spaces.

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper does not release a pretrained generative model, scraped dataset, or other high-risk asset requiring controlled release safeguards.

12. **Licenses for existing assets**

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

    Answer: [No]

    Justification: The original datasets, baseline methods, and rasterizer are cited, and Section I describes their academic use without redistribution. The paper does not itemize the license and terms of use of every asset.

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper introduces a method and describes its CUDA implementation, but does not release a new dataset or pretrained model.

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

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

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject experiments, so IRB approval is not applicable.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs are not a component of the proposed representation, conditioning equations, training objective, or evaluation method.