3D Skew-Normal Splatting

Xiangru Wu
Fudan University
xiangruwu22@m.fudan.edu.cn

Ke Fan
Fudan University
kfan21@m.fudan.edu.cn

Yanwei Fu
Fudan University
yanweifu@fudan.edu.cn

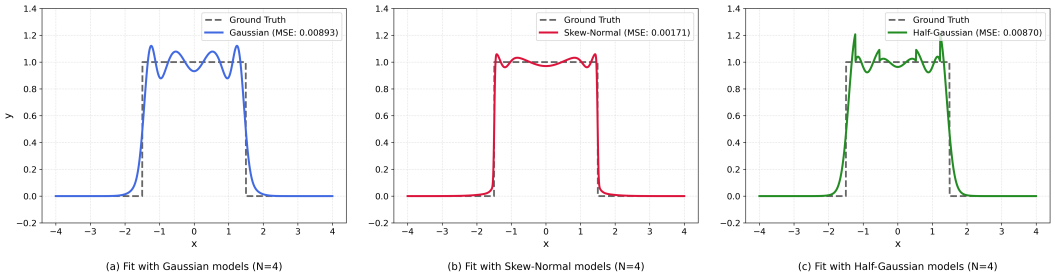
Abstract

3D Gaussian Splatting (3DGS) has emerged as a leading representation for real-time novel view synthesis and has been widely adopted in various downstream applications. The core strength of 3DGS lies in its efficient kernel-based scene representation, where Gaussian primitives provide favorable mathematical and computational properties. However, under a finite primitive budget, the symmetric shape of each primitive directly affects representation compactness, especially near asymmetric structures such as object boundaries and one-sided surfaces. Recent works have explored more complex kernel distributions; however, they either remain within the elliptical family or rely on hard truncation, which limits continuous shape control and introduces distributional discontinuities. In this paper, we propose Skew-Normal Splatting (SNS), which adopts the Azzalini Skew-Normal distribution as the fundamental primitive. By introducing a learnable and bounded skewness parameter, SNS can continuously interpolate between symmetric Gaussians and Half-Gaussian-like shapes, enabling flexible modeling of both sharp boundaries and interior regions. Moreover, SNS preserves analytical tractability under affine transformations and marginalization. This property allows seamless integration into existing Gaussian Splatting rasterization pipelines. Furthermore, to address the strong coupling between scale, rotation, and skewness parameters, we introduce a decoupled parameterization and a block-wise optimization strategy to enhance training stability and accuracy. Extensive experiments on standard novel-view synthesis benchmarks show that SNS consistently improves reconstruction quality over Gaussian and recent non-Gaussian kernels, with clearer benefits on sharp boundaries and thin or one-sided structures.

1 Introduction

In the field of novel view synthesis, the increasing demand for real-time interactivity has exposed the limitations of traditional NeRF-based [1, 2, 3, 4] methods regarding rendering efficiency and training overhead. Since its introduction, 3D Gaussian Splatting [5] has rapidly emerged as a dominant alternative to early NeRF approaches. By leveraging explicit scene representations and fast, differentiable rendering, it achieves a superior balance between visual fidelity and computational performance and has been widely used in fields such as robotics [6], autonomous driving [7], and cultural heritage preservation [8]. The success of 3D Gaussian Splatting is attributed to its use of Gaussian functions as kernels, which provide powerful approximation capabilities alongside mathematical and computational efficiency [9, 10]. Nevertheless, the universal approximation property of kernel-based methods remains largely a theoretical guarantee, typically predicated on the availability of an infinite and sufficiently dense distribution of kernels. In the practical regime of a finite primitive budget, the distributional form of each kernel directly dictates representation compactness and reconstruction accuracy. Symmetric Gaussian kernels often require multiple overlapping primitives to approximate sharp boundaries, leading to redundant representations and blurred reconstructions.

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



(a) Fit with Gaussian models (N=4) (b) Fit with Skew-Normal models (N=4) (c) Fit with Half-Gaussian models (N=4)

Figure 1: **Reconstruction results.** We fit a square wave with different basis models using a sum of four primitives from: **(a)** Gaussian, **(b)** Skew-Normal, and **(c)** Half-Gaussian distributions. Their parameters are optimized via gradient descent. Our Skew-Normal model achieves the lowest Mean-Squared Error (MSE), demonstrating its better capability in approximating signals with sharp discontinuities.

Several splatting methods address this limitation by giving primitives more expressive splatting kernels. DRK [11] constructs flexible planar contours using learnable radial bases, L_1/L_2 distance blending, and edge sharpening. DisC-GS [12] clips projected Gaussians with Bézier-defined binary masks to model hard boundaries and uses approximate gradients for optimization. Another line of work explores alternative distributions for splatting primitives. Generalized Exponential Splatting (GES) [13] and Student Splatting and Scooping (SSS) [14] propose to extend the Gaussian primitive by modifying the radial density post-whitening; however, such methods are still restricted to the elliptical distribution family, which imposes a persistent radial symmetry on the representation. Among these efforts, HGS [15] adopts the Half-Gaussian distribution, representing an early attempt to introduce asymmetric kernels into scene modeling. However, this approach relies on a hard truncation scheme that simply divides a Gaussian distribution at its centroid, thereby offering only limited expressivity in terms of skewness. Furthermore, such truncation inherently introduces discontinuities into the primitive's distribution.

To further characterize the asymmetry that is prevalent in real data, many works in Statistics [16, 17, 18, 19, 20, 21] extend the Gaussian distribution with additional shape parameters to capture skewness. These distributions are widely used in fields such as economics and finance, biomedicine, and spatial statistics [22, 23, 24, 25, 26]. Among these works, Azzalini's Skew-Normal distribution [27] is widely regarded as a classical extension of the Gaussian distribution, due to its simplicity and clear probabilistic construction, providing a useful reference for our work.

To this end, we propose Skew-Normal Splatting (SNS), which employs the Skew-Normal distribution of Azzalini as the underlying primitive for scene modeling. By subsuming the standard Gaussian and half-Gaussian distributions as special or limiting cases, the Skew-Normal distribution provides a continuous and tunable mechanism for asymmetric modeling. On one hand, it offers the flexibility to capture non-symmetric structures such as sharp edges and corners; on the other, it maintains the capacity to effectively represent the interior regions of objects. As Fig. 1 shows, we fit the same 1D square wave using Gaussian, Skew-Normal, and Half-Gaussian kernels, respectively. For a fair comparison, all methods are restricted to 4 components and optimized via gradient descent. The Gaussian model exhibits boundary ringing-like artifacts, including overshoot and undershoot near step transitions. The Half-Gaussian method offers a marginal improvement but introduces discontinuities and sharp peaks at the step changes. In contrast, SNS can effectively fit both sharp edges and flat portions of the square wave, demonstrating its superior representational capacity.

Furthermore, the SNS remains analytically tractable under affine transformations and marginalization. Leveraging this mathematical convenience, we re-derive the projection and integration routines required to incorporate SNS into the splat-based rasterization pipeline, thereby enabling highly efficient computation.

Beyond its representational benefits, the transition to asymmetric primitives introduces new analytical and optimization challenges. We observe that the location parameter of a Skew-Normal distribution does not coincide with its spatial mean. We thus derive a corrected operational center to ensure accurate bounding-box localization. Furthermore, to resolve the optimization instability caused by the coupling between orientation and skewness, we introduce a decoupled parameterization in the primitive's canonical coordinate system, supported by a block-wise alternating update strategy.

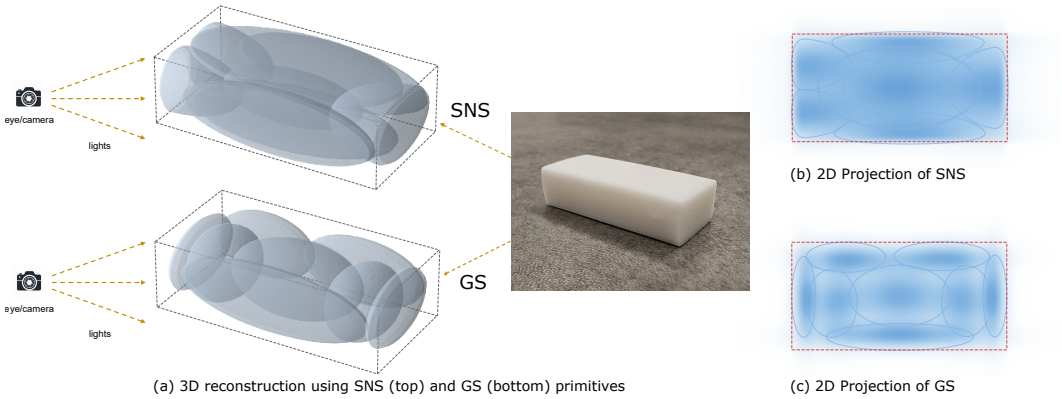


Figure 2: **Comparison between Skew-Normal and Gaussian primitives.** Standard Gaussian primitives are symmetric and often require multiple overlapping kernels to approximate sharp or one-sided structures. By contrast, SNS introduces a learnable skewness parameter, enabling diverse asymmetric primitives to accurately fit sharp corners and flat surfaces.

This design effectively reduces parameter interference, ensuring stable convergence and improved generalization. Formally, our contributions include:

- We introduce Azzalini's Skew-Normal distribution to Gaussian Splatting, which has varying degrees of skewness for 3D scene modeling and can simultaneously handle object boundaries and interiors.
- We derive a complete analytical framework for Skew-Normal kernels, including closed-form affine projection, corrected footprint localization, and efficient gradient computation for rasterization, preserving compatibility with existing 3DGS pipelines.
- We identify the optimization coupling between geometric and skewness parameters and propose a decoupled reparameterization along with a block-wise alternating update scheme to enhance training stability.
- SNS achieves consistent PSNR gains over recent non-Gaussian variants and improves visual preservation of sharp structures and object completeness.

2 Related Work

NeRF methods. Early approaches to 3D reconstruction such as SfM [28, 29] and MVS [30, 31] typically rely on explicit geometric estimation. Recently, Neural Radiance Fields (NeRF) [1, 2] implicitly learn scene geometry and appearance by parameterizing volumetric density and color through neural networks, utilizing volume rendering to synthesize novel views. Mip-NeRF [3] and Mip-NeRF 360 [4] reduce aliasing and extend NeRF to unbounded scenes, while explicit or hybrid representations such as Plenoxels [32] and K-Planes [33] improve training and rendering efficiency. Beyond reconstruction, NeRF has been extended to sparse-view synthesis [34, 35], object-centric decomposition [36, 37], neural surface reconstruction [38, 39], and dynamic scene modeling [40, 41], substantially advancing neural 3D representations. Despite their strong rendering quality, NeRF-based methods typically rely on dense ray sampling and implicit network evaluation, which makes training and rendering computationally expensive compared with explicit splatting-based representations.

Splatting methods. The 3DGS [5] models the scene with optimizable Gaussian elements and projects them onto the image plane through an efficient splatting rasterizer, thereby significantly accelerating the rendering process. Building upon this paradigm, several works have improved 3DGS across multiple dimensions. Mip-Splatting [42] focuses on anti-aliasing and multi-scale rendering, while Scaffold-GS [43] and Octree-GS [44] introduce hierarchical structures to enhance representation efficiency. To optimize the growth and pruning strategies of Gaussian primitives, Pixel-GS [45] and AbsGS [46] refine the density control mechanisms. Furthermore, 3DGS has been extended to a broader range of applications: GaussianEditor [47, 48] enables scene editing within the Gaussian representation, and 4DGS [49, 50] further generalizes the framework to dynamic scene modeling.

Collectively, these advancements have established 3DGS as a powerful alternative for efficient 3D scene representation and rendering. Beyond these domains, the framework has also been applied to 3D content generation [51, 52, 53, 54], language-driven scene understanding [55, 56, 57], and 3D scene segmentation [58, 59]. More recently, several works have explored extending 3DGS by modifying the underlying kernel functions of the Gaussian primitives. For instance, GES [13] adopts the Generalized Exponential Function, while SSS employs the Student's t-distribution [14]. Pushing this further, 3D-HGS [15] introduces the half-Gaussian distribution to facilitate the representation of more irregular primitives. Unlike GES and SSS, which modify the radial profile after whitening and therefore remain largely symmetric in canonical space, our SNS introduces directional skewness as an intrinsic primitive parameter. Unlike 3D-HGS, SNS avoids hard truncation and provides a continuous path from Gaussian-like to half-Gaussian-like shapes.

3 Method

3.1 Gaussian Splatting

Scene Representation. 3DGS models the scene using 3D Gaussian kernels. Each kernel is parameterized by a center position $\mu \in \mathbb{R}^3$, a 3D covariance matrix $\Sigma \in \mathbb{R}^{3\times3}$, an opacity value $o \in [0, 1]$, and view-dependent color features c. To ensure positive semi-definiteness during optimization, Σ is factorized into a scaling matrix $S \in \mathbb{R}^{3\times3}$ and a rotation matrix $R \in \mathbb{R}^{3\times3}$, formulated as: $\Sigma = RSS^\top R^\top$. These 3D Gaussians are projected onto the 2D screen space to render images. Given a viewing transformation matrix W and the Jacobian of the affine projection approximation J, the 2D covariance matrix, denoted by Σ', can be computed as: $\Sigma' = JW\Sigma W^\top J^\top$.

Volumetric Alpha-Blending. The rendering process evaluates the final color $C(u)$ at pixel u through a tile-based rasterizer. Analogous to volumetric rendering, 3DGS performs alpha-blending by sorting the projected Gaussians along the ray corresponding to the pixel from front to back. For a set of $\mathcal{N}$ ordered, intersecting Gaussians, the pixel color is aggregated as [5]:

$$C(u) = \sum_{i=1}^{\mathcal{N}} c_i a_i(u) \prod_{j=1}^{i-1} (1 - a_j(u)), \tag{1}$$

where c_i is the color evaluated from the SH coefficients. The term $a_i(u)$ denotes the spatial influence of the i-th Gaussian evaluated at pixel u, which is obtained by multiplying the learned opacity o_i with the 2D Gaussian kernel:

$$a_i(u) = o_i \exp\left(-\frac{1}{2}(u - \hat{\mu}_i)^\top \hat{\Sigma}_i^{-1}(u - \hat{\mu}_i)\right), \tag{2}$$

where $\hat{\mu}_i$ is the projected 2D center position.

3.2 3D Skew-Normal Kernel

We inherit Azzalini's definition of the Skew-Normal Distribution (SND), which provides directional asymmetry through a new parameter α. A d-dimensional SND random variable Y is denoted as $Y \sim \mathcal{SN}_d(\mu, \Omega, \alpha)$, where the parameters are called location, scale matrix, and slant [27], respectively. The slant parameter α is highly coupled with Ω. When α is kept fixed, applying a spatial rotation to the model does not only reorient the primitive but also changes its intrinsic geometric shape. To decouple the shape from the orientation, we propose an unnormalized and reparameterized 3D Skew-Normal kernel, which is defined by a location vector $\mu \in \mathbb{R}^3$, a scale matrix $\Omega = RSS^\top R^\top \in \mathbb{R}^{3\times3}$, and a learnable latent variable $k \in \mathbb{R}^3$:

$$SN(x; \mu, \Omega, k) = G_3(x - \mu; \Omega)\, \Phi\left(\alpha^\top \omega^{-1}(x - \mu)\right), \tag{3}$$

where $\alpha = \omega RS^{-1}k$, $\omega = (\Omega \odot \mathbf{I}_3)^{1/2}$, $G_3(x - \mu; \Omega) = \exp\left(-\frac{1}{2}(x - \mu)^\top \Omega^{-1}(x - \mu)\right)$, and $\Phi(\cdot)$ denotes the cumulative distribution function (CDF) of the standard univariate normal distribution. The symbol $\odot$ refers to the Hadamard product. This formulation provides four main advantages:

Continuous Shape Interpolation. The Skew-Normal distribution includes the Gaussian and half-Gaussian as limiting cases, providing a natural continuum between symmetric and maximally skewed geometries. It reduces to a Gaussian ellipsoid when $\alpha = \mathbf{0}$, and approaches a half-ellipsoid as

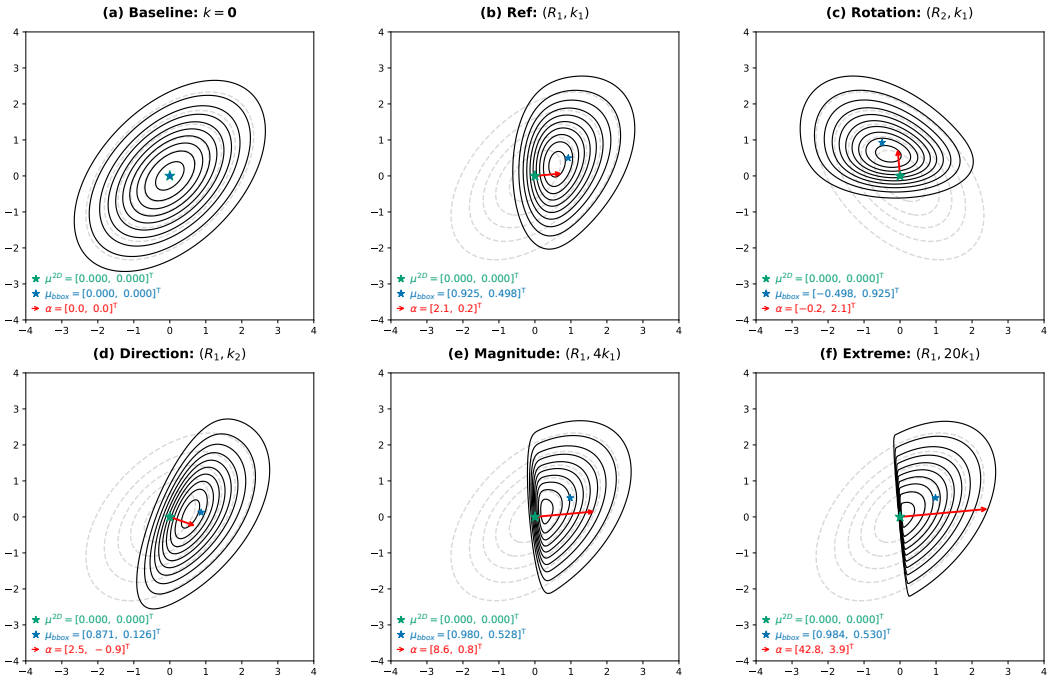


Figure 3: **Parameter decoupling of bivariate SND.** These panels demonstrate the continuous shape interpolation and decoupled control of the Skew-Normal density. **(a)** Baseline standard symmetric Gaussian. **(b vs. d)** Directional control via the orientation of k. **(b, e, f)** Magnitude variation and rapid convergence. **(b vs. c)** Intrinsic nature demonstrated via rigid global rotation R. In each panel, the dashed line indicates the symmetric Gaussian base and the solid lines depict the Skew-Normal density. The green star marks the projected location μ^{2D}, the blue star marks the corrected bounding-box center μ_{bbox}, and the red arrow indicates the slant vector α. All panels share an identical scaling matrix S.

$\|\alpha\| \to \infty$. As illustrated in Fig. 3, this continuous interpolation enables a single kernel to represent a broad spectrum of geometric structures. By increasing the magnitude from $\|k_1\| \approx 2.24$ to $4\|k_1\| \approx 8.94$, the kernel reshapes into a sharp half-ellipsoid. Crucially, further scaling to $20\|k_1\|$ yields a slight visual change, demonstrating that the skewness converges rapidly to its limiting form. Such flexibility extends the expressive power of Gaussian kernels while preserving a compact and principled parameterization.

Bounded Skewness. The shape of the SND is influenced by the slant α and the scale matrix Ω. To provide a single, scalar measure of skewness, Mardia's measure of skewness [27] is expressed as:

$$\gamma_{1,d}^M = \left(\frac{4-\pi}{2}\right)^2 \left(\frac{(2/\pi)\alpha_*^2}{1+(1-2/\pi)\alpha_*^2}\right)^3, \tag{4}$$

where $\alpha_* = \left(\alpha^\top \bar{\Omega} \alpha\right)^{\frac{1}{2}} = \|k\|_2 \in [0, \infty)$. As shown in Eq.(4), $\gamma_{1,d}^M$ is monotonically increasing with respect to α_*. When $\alpha_* = 0$, $\gamma_{1,d}^M = 0$; when $\alpha_* \to \infty$, $\gamma_{1,d} \to \frac{2(\pi-4)^2}{(\pi-2)^3}$, which is approximately 0.99067. Therefore, the skewness of the SND is confined to a bounded region, preventing the distribution from being excessively stretched. This bounded property is important for the 3DGS framework. 3DGS relies on tile-based rasterization, which requires primitives to have a spatially compact footprint for efficient bounding box calculation. An unbounded asymmetric tail would cause a primitive to intersect with an excessive number of tiles, degrading rendering performance. On the other hand, the local affine approximation (via Jacobian J) employed in 3DGS is fundamentally predicated on a localized spatial footprint. An arbitrarily extended tail would violate this assumption and lead to instability during optimization.

Decoupled Skewness Measure. By defining the slant parameter as $\alpha = \omega R S^{-1} k$, we decouple the asymmetry of the distribution from its affine transformation components (i.e., R and S). Under this

reparameterization, the term α_* in Eq.(4) reduces to the L_2-norm of the skew vector k, implying that $\gamma^M_{1,d}$ is solely determined by $\|k\|_2$. It reveals that the magnitude of k comprehensively encapsulates the departure from normality. In Fig. 3, panels (b) and (c) share the same k_1 but differ in rotation R. The resulting density in (c) is a rigid rotation of (b), verifying that k is an intrinsic property defined in the primitive's canonical coordinate system; varying R only alters global orientation without deforming the asymmetric kernel itself. Additionally, shifting k's orientation (e.g., rotating $k_1 = (2, -1)^\top$ to k_2 while maintaining $\|k_1\| = \|k_2\|$) dictates the specific axis of the skew, remapping the kernel's asymmetry to a new direction.

Closed-form 2D Projection. The SND is closed under affine transformations. We derive the analytical form by projecting a 3D primitive to the 2D image plane. This derivation requires an affine transformation matrix $A = JW \in \mathbb{R}^{2\times 3}$. Formally, a 2D Skew-Normal kernel is described by:

$$SN^{2D}(u; \mu^{2D}, \Omega^{2D}, m^{2D}) = G_2(u - \mu^{2D}; \Omega^{2D})\, \Phi\left((m^{2D})^\top (u - \mu^{2D})\right). \tag{5}$$

where $u \in \mathbb{R}^2$ denotes the pixel coordinate,

$$\begin{aligned} \mu^{2D} &= A\mu, \quad \Omega^{2D} = A\Omega A^\top, \\ m^{2D} &= \frac{(\Omega^{2D})^{-1} ARSk}{\sqrt{1 + k^\top k - (ARSk)^\top (\Omega^{2D})^{-1} ARSk}}. \end{aligned} \tag{6}$$

3.3 Forward Modeling of Splatting

Restricting primitives strictly to a positive density space bounds their representational capacity. Following SSS, we allow signed opacity values $o_i \in [-1, 1]$, enabling primitives to both contribute color and subtract volumetric density from the scene. This subtractive capability can represent complex topologies with fewer kernels. At a pixel u, the front-to-back alpha blending evaluates as:

$$C(u) = \sum_{i\in\mathcal{N}} c_i o_i SN_i^{2D}(u) \prod_{j=1}^{i-1} \left(1 - o_j SN_j^{2D}(u)\right), \tag{7}$$

where $\mathcal{N}$ is the depth-sorted set of intersecting kernels.

Traditional splatting frameworks, such as 3DGS and SSS, anchor their tile-based bounding boxes to the nominal location parameter μ^{2D}, which is the mean of the distribution. As shown in Fig. 3, skewness shifts the probability mass away from μ^{2D}. While rasterization still uses a symmetric bounding region, centering bounding boxes at μ^{2D} is ineffective in capturing the spatial density of skewed objects. To address this, we shift the center of the bounding region to the projected mean, deriving the corrected operational center, μ_{bbox}, as a deterministic spatial offset:

$$\mu_{bbox} = \mu^{2D} + \sqrt{\frac{2}{\pi}} \cdot \frac{ARSk}{\sqrt{1 + k^\top k}}. \tag{8}$$

3.4 Backward Propagation

Our backward pass follows the two-stage structure of standard 3DGS, with a rendering backward pass and a preprocessing backward pass. To accommodate the skew-normal kernel, we revise the gradient propagation path and adapt the computations in both stages. More details are in appendix B.

Gradient Paths under Reparameterization. The reparameterization of α is essential to simplifying the backward pass. Without it, the projected slant takes the nested form:

$$m^{2D} = \frac{(\Omega^{2D})^{-1} A\omega\bar{\Omega}\alpha}{\sqrt{1 + \alpha^T\bar{\Omega}\alpha - (A\omega\bar{\Omega}\alpha)^T (\Omega^{2D})^{-1} A\omega\bar{\Omega}\alpha}}, \tag{9}$$

where α is the original slant parameter, $\bar{\Omega} = \omega^{-1}\Omega\omega^{-1}$, and (A, Ω^{2D}, ω) are defined in Eq. (6) and Eq. (3). Since both rotation and scale determine Ω^{2D}, their gradients must propagate through the dependency: $(R, S) \to \Omega \to \omega, \bar{\Omega}, (\Omega^{2D})^{-1} \to m^{2D}$. This leads to a more coupled gradient flow. As we introduce $\alpha = \omega RS^{-1}k$ and $Q = ARS$, the propagation can be reduced to a much more compact form: $(R, S) \to Q \to m^{2D}$, while preserving exactly the same skew-normal distribution.

Rasterization gradients. In 3DGS, the rendering backward pass accumulates gradients for the projected mean μ^{2D} and conic matrix $(\Omega^{2D})^{-1}$ through every pixel covered by a primitive; the preprocessing backward pass then propagates them to the 3D parameters. We follow the same division for SNS. To avoid repeating primitive-dependent calculations at every pixel, we express the derivatives of SN^{2D} with respect to μ^{2D}, k, and Q in terms of four shared auxiliary terms. Let $u \in \mathbb{R}^2$ denote a pixel coordinate and define $\Delta = u - \mu^{2D}$, $G_2 = \exp\big(-\frac{1}{2}\Delta^T(\Omega^{2D})^{-1}\Delta\big)$, $v_\Delta = (\Omega^{2D})^{-1}\Delta$, and $\lambda = (m^{2D})^T\Delta$. The auxiliary terms are

$$\begin{aligned} c_\Delta &= \Phi \cdot G_2 \cdot v_\Delta, \\ M_\Delta &= \Phi \cdot G_2 \cdot \big(v_\Delta v_\Delta^T\big), \\ s_\Delta &= \frac{G_2}{\sqrt{2\pi}} \cdot \exp\left(-\frac{1}{2}\lambda^2\right), \\ p_\Delta &= \frac{G_2}{\sqrt{2\pi}} \cdot \exp\left(-\frac{1}{2}\lambda^2\right) \cdot v_\Delta, \end{aligned} \tag{10}$$

where Φ denotes the standard normal CDF evaluated at λ. During rendering, we weight each auxiliary term by the upstream gradient $\partial L/\partial SN^{2D}$ and accumulate it over the pixels covered by the primitive. Then, in the preprocessing pass, we combine the accumulated values and propagate the resulting gradients using the chain rule.

3.5 Stable Optimization via Parameter Decoupling

Optimizing Skew-Normal kernels is particularly challenging due to the coupling between the geometric and skewness parameters. We update the positional parameters μ using Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) [14], which introduces stochasticity to help the model escape local minima. The greater challenge, however, comes from the slant parameter α, as it deeply couples the rotation R, scaling S, and the latent variable k. We decouple them from two perspectives:

Internal parameter dependencies. In Sec. 3.2, the magnitude $\|k\|_2$ determines Mardia's measure of skewness, while the normalized vector $k/\|k\|_2$ determines the skew direction. As shown in Fig. 3 and Eq. (4), the shape and Mardia's skewness measure rapidly approach their limiting values as increases. Bounding skew magnitude can avoid spending optimization capacity in an extreme-skew regime where further increases produce little shape change. We therefore factorize k into a bounded magnitude scalar $m_k \in \mathbb{R}_+$ and a direction vector $d_k \in \mathbb{R}^3$:

$$m_k = \frac{8}{1 + exp(-\frac{x}{6})}, \quad d_k = \frac{v}{\|v\|_2}, \tag{11}$$

where $x \in \mathbb{R}, v \in \mathbb{R}^3$. The constant 8 sets the maximum skew magnitude, while 6 controls the slope of the mapping, with a larger value giving a smoother transition.

Inter-parameter interference. Although decomposing k into magnitude and direction improves stability, updating shape parameters (R, S, m_k, d_k) simultaneously still leads to suboptimal convergence. Specifically, adjustments to the base ellipsoid's orientation frequently conflict with the skew direction. To address this, we divide the training process into two phases. First, all parameters (R, S, m_k, d_k) are jointly optimized using standard Adam to establish a reasonable global structure. In the second phase, a Block Coordinate Descent (BCD) strategy is activated. The optimization follows a cyclic alternating schedule: in the first part of each cycle, the ellipsoidal base parameters (R, S) are updated while the skew parameters (m_k, d_k) are frozen; in the second part, (m_k, d_k) are refined while (R, S) are held constant.

While SGHMC is utilized for the positional parameters μ, the Adam optimizer is employed for all remaining variables. SGHMC naturally provides stochastic global search, preventing positional parameters from collapsing into local minima. Meanwhile, the alternating update scheme governs the shape parameters, ensuring geometric stability throughout training.

4 Experiments

Datasets and Metrics. We evaluate SNS on 11 scenes across 3 datasets to validate our contributions: 7 scenes from Mip-NeRF 360 [4], 2 scenes from Tanks & Temples [60], and 2 scenes from Deep

Table 1: **Quantitative comparison with baselines.** The best and second-best results are highlighted in dark red and light red, respectively.

Dataset	Mip-NeRF360			Tanks&Temples			Deep Blending		
Method	PSNR ↑	SSIM↑	LPIPS↓	PSNR ↑	SSIM↑	LPIPS↓	PSNR ↑	SSIM↑	LPIPS↓
Mip-NeRF	29.23	0.844	0.207	22.22	0.759	0.257	29.40	0.901	0.245
3DGS	28.69	0.870	0.182	23.14	0.841	0.183	29.41	0.903	0.243
GES	28.72	0.794	0.250	23.34	0.836	0.198	29.82	0.901	0.252
3DHGS	29.56	0.873	0.178	24.50	0.857	0.169	29.76	0.905	0.242
Fre-GS	27.85	0.826	0.209	23.96	0.841	0.183	29.93	0.904	0.240
Scaffold-GS	28.84	0.848	0.220	23.96	0.853	0.177	30.21	0.906	0.254
3DGS-MCMC	29.89	0.900	0.190	24.29	0.860	0.190	29.67	0.890	0.320
SSS	29.90	0.893	0.145	24.87	0.873	0.138	30.07	0.907	0.247
Ours	30.17	0.895	0.144	25.08	0.877	0.132	30.30	0.912	0.244

Blending [61]. Following the standard evaluation metrics, we employ three metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS). We report dataset-level averages, with per-scene results in the appendix.

Baselines. (1) kernel-level variants such as GES [13], 3D-HGS [15] and SSS [14], which introduce non-Gaussian primitives; (2) optimization-focused methods like Scaffold-GS [43], Fre-GS [62] and 3DGS-MCMC [63]; and (3) Neural Radiance Field method such as Mip-NeRF [3]. For a fair comparison, we use the results either from original papers or run publicly available implementations under the same settings.

Implementation Details. Our model is implemented in PyTorch, with the core rasterization engine extended from the 3DGS framework to support the asymmetric 2D Skew-Normal kernel. All experiments are conducted on NVIDIA RTX A6000 GPUs. In terms of computational efficiency, SNS demonstrates a training overhead comparable to SSS; for instance, in the drjohnson scene, SNS completes 30,000 iterations in 38m 4s (29.89 dB PSNR), while SSS requires 38m 7s (29.66 dB PSNR) on the same test set.

4.1 Results

Performance Comparison.

We report the quantitative comparison of our method with state-of-the-art methods in Tab. 1. Overall, our method achieves the best or highly competitive results across all datasets and metrics, validating the effectiveness and robustness of the proposed approach. Our method showed the most consistent and strongest performance on PSNR, achieving 30.17 on Mip-NeRF360, 25.08 on Tanks&Temples, and 30.30 on Deep Blending. The improvement is most notable on Tanks&Temples. Compared with SSS, the strongest baseline, our method achieves higher PSNR (24.87→25.08) and SSIM (0.873→0.877), and lower LPIPS (0.138→0.132) on Tanks&Temples. SNS remains highly competitive on SSIM for Mip-NeRF360 and LPIPS for Deep Blending, achieving results nearly on par with the best-performing methods.

Qualitative Results. Our method demonstrates better object completeness across multiple scenes in Fig. 4. For locally complex objects, such as the curtain above the window in the *room* scene and the tower-shaped structure in the *train* scene, our method maintains the complete structure and clear boundaries. In contrast, SSS and 3DGS-MCMC exhibit noticeable missing regions in the *train* scene, while GS and 3DHGS produce holes and fuzzy boundaries in the *room* scene.

In terms of texture clarity, our method more effectively preserves high-frequency details. For example, the wavy stripes on top of the transparent plastic box in the *bonsai* scene, the texture and shadows on the cloth behind the chair in the *kitchen* scene, and the distant trees in the *train* scene are all reconstructed and rendered with clearer details. By contrast, the original GS produces over-smoothed reconstructions in texture-rich regions, leading to noticeable blur. Besides, in the *drjohnson* scene, SNS exhibits the mildest artifacts among all compared methods. These results show that our method better handles complex regions in novel view synthesis.



(a) kitchen (b) bonsai (c) room (d) train (e) drjohnson

Figure 4: **Reconstruction quality.** We visualize five representative scenes, highlighting and zooming in on regions exhibiting the most prominent differences for comparison.

Table 2: **Per-scene PSNR comparison of various reconstruction kernels.** We provide a detailed PSNR evaluation of the methods using different base kernels across the Mip-NeRF 360, Deep Blending, and Tanks&Temples datasets. The best and second-best results are highlighted in dark red and light red, respectively.

Method	Mip-NeRF 360							Tanks&Temples		Deep Blending	
	Bicycle	Bonsai	Counter	Garden	Kitchen	Room	Stump	Train	Truck	Playroom	DrJohnson
3DGS	25.25	31.98	28.70	27.41	30.32	30.63	26.55	21.09	25.18	30.04	28.77
GES	24.87	31.97	28.75	27.07	31.07	31.17	26.17	21.73	24.94	30.29	29.35
3D-HGS	25.39	33.30	29.62	27.68	32.17	32.12	26.64	22.95	26.04	30.20	29.32
SSS	25.68	33.50	29.87	28.09	32.43	32.57	27.17	23.32	26.41	30.47	29.66
SNS (Ours)	25.85	33.70	30.27	28.22	32.81	33.02	27.31	23.61	26.56	30.71	29.89

Different kernels comparison. Table 2 provides the detailed PSNR scores evaluated with GS, GES, HGS, SSS, and our SNS. In this comparison, we treat standard 3DGS as the baseline. Across all scenes, SNS outperforms all other models by at least 0.1 dB. For example, our method outperforms SSS by 0.45 dB on the Room scene, 0.29 dB on the Train scene, and 0.24 dB on the Playroom scene. Overall, our SN formulation consistently delivers the best performance across all tested 3D scenes.

4.2 Ablations

Table 3 details the ablation study on individual module contributions. Using the vanilla Skew-Normal distribution as our baseline, we compare three configurations: **(1)** k**-decomposition**: Decomposes

Table 3: **Summary ablation across datasets.** We present the average performance of our Full SNS compared with its ablated versions across three main datasets. The best and second-best results are highlighted in dark red and light red, respectively.

Dataset	Mip-NeRF360			Tanks&Temples			Deep Blending		
Model Variants	PSNR ↑	SSIM↑	LPIPS↓	PSNR ↑	SSIM↑	LPIPS↓	PSNR ↑	SSIM↑	LPIPS↓
Vanilla SNS	30.0643	0.8942	0.1443	25.0388	0.8765	0.1324	30.1581	0.9097	0.2448
k-decomposition	30.1204	0.8950	0.1442	25.0090	0.8768	0.1319	30.1185	0.9118	0.2441
Alternating Optimization	30.1390	0.8949	0.1445	25.0328	0.8776	0.1322	30.2969	0.9122	0.2436
Full SNS	30.1700	0.8954	0.1445	25.0839	0.8775	0.1325	30.3034	0.9119	0.2444

the skew vector k into magnitude and direction, without alternating optimization. **(2) Alternating optimization**: Alternately optimizes (R, S) and k, without k-decomposition. **(3) Full model**: Incorporates both k-decomposition and alternating optimization. Full experimental results are available in Appendix C.

Both modules contribute to the final performance through distinct mechanisms. Alternating optimization yields consistent improvements in pixel-level reconstruction, increasing mean PSNR from 30.06 to 30.14 on Mip-NeRF 360 and from 30.16 to 30.30 on Deep Blending. This indicates that alternating updates for (R, S) and k effectively decouple geometric parameters from skew parameters, stabilizing the optimization process. Meanwhile, k-decomposition consistently enhances SSIM and LPIPS across all datasets, suggesting that decomposing the skew vector into magnitude and direction provides a superior parameterization that improves optimization dynamics. The full model delivers the most significant gains in PSNR, while maintaining highly competitive LPIPS and SSIM performance across scenes. Thus, these results suggest that the two optimization components provide complementary but moderate gains. The full model achieves the best average PSNR across all datasets, while individual variants may slightly outperform it on perceptual metrics in some cases, indicating that the modules mainly improve reconstruction fidelity and stability rather than uniformly optimizing every metric.

5 Conclusion

We propose Skew-Normal Splatting, which introduces controllable skewness into Gaussian Splatting to better represent asymmetric geometries, sharp boundaries, and one-sided structures. SNS preserves analytical projection for efficient splatting and further improves optimization stability through decoupled parameterization and block-wise updates. Experiments show consistent improvements over existing Gaussian and non-Gaussian kernels.

6 Limitations and future work.

While SNS improves the reconstruction of asymmetric geometries and sharp boundaries, it introduces certain limitations. First, evaluating the Skew-Normal distribution requires computing the Cumulative Distribution Function (CDF). In our current implementation, we directly employ the highly accurate numerical solution (the **erf** function) to ensure mathematical precision. While faster analytic approximations exist, relying on the precise **erf** incurs moderate penalties on rendering speed. Second, as SNS primarily enhances geometric representation, it still shares the limitations of 3DGS in modeling highly specular surfaces using Spherical Harmonics (SH). Third, as observed in our ablation studies, while the full combined model achieves the highest average PSNR, individual variants slightly outperform it on SSIM and LPIPS. This reveals that a consistently optimal optimization method that uniformly maximizes all evaluation metrics has not been found.

Future work will focus on reducing the CDF computational overhead and exploring more comprehensive optimization algorithms to jointly improve both reconstruction fidelity and perceptual quality. Furthermore, rather than applying complex asymmetric kernels globally, we plan to integrate SNS with spatial hierarchical structures. This would allow the model to adaptively allocate Skew-Normal primitives along high-frequency object boundaries while falling back to simpler Gaussians in smooth regions, thereby optimizing the overall efficiency-to-quality trade-off.

References

[1] 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 Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, Lecture Notes in Computer Science, pages 405–421. Springer, 2020.

[2] 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. *Commun. ACM*, 65(1):99–106, 2022.

[3] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 5835–5844. IEEE, 2021.

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 5460–5469. IEEE, 2022.

[5] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.*, 42(4):139:1–139:14, 2023.

[6] Nikhil Varma Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian A. Scherer, Deva Ramanan, and Jonathon Luiten. SplaTAM: Splat, Track & Map 3D Gaussians for Dense RGB-D SLAM. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 21357–21366. IEEE, 2024.

[7] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. DrivingGaussian: Composite Gaussian Splatting for Surrounding Dynamic Autonomous Driving Scenes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 21634–21643. IEEE, 2024.

[8] Mahtab Dahaghin, Myrna C. Silva, Kourosh Riahidehkordi, Matteo Toso, and Alessio Del Bue. Gaussian Heritage: 3D Digitization of Cultural Heritage with Integrated Object Segmentation. *CoRR*, abs/2409.19039, 2024.

[9] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001.

[10] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization & Computer Graphics*, 8(03):223–238, 2002.

[11] Yi-Hua Huang, Ming-Xian Lin, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Deformable Radial Kernel Splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025*. IEEE, 2025.

[12] Haoxuan Qu, Zhuoling Li, Hossein Rahmani, Yujun Cai, and Jun Liu. DisC-GS: Discontinuity-aware Gaussian Splatting. In *Advances in Neural Information Processing Systems*, volume 37, pages 112284–112309. Neural Information Processing Systems Foundation, Inc., 2024.

[13] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 19812–19822. IEEE, 2024.

[14] Jialin Zhu, Jiangbei Yue, Feixiang He, and He Wang. 3D Student Splatting and Scooping. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 21045–21054. Computer Vision Foundation / IEEE, 2025.

[15] Haolin Li, Jinyang Liu, Mario Sznaier, and Octavia I. Camps. 3D-HGS: 3D Half-Gaussian Splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 10996–11005. Computer Vision Foundation / IEEE, 2025.

[16] Debasis Kundu. Geometric Skew Normal Distribution. *Sankhya B*, 76:167 – 189, 2014.

[17] David Elal-Olivero. ALPHA-SKEW-NORMAL DISTRIBUTION. *Proyecciones (antofagasta)*, 29:224–240, 2010.

[18] Valentina Mameli and Monica Musio. A Generalization of the Skew-Normal Distribution: The Beta Skew-Normal. *Communications in Statistics - Theory and Methods*, 42:2229 – 2244, 2011.

[19] Govind S. Mudholkar and Alan David Hutson. The epsilon-skew-normal distribution for analyzing near-normal data. *Journal of Statistical Planning and Inference*, 83:291–309, 2000.

[20] Reinaldo Boris Arellano-Valle and Adelchi Azzalini. On the Unification of Families of Skew-normal Distributions. *Scandinavian Journal of Statistics*, 33, 2006.

[21] Barry C Arnold and Robert J Beaver. Hidden truncation models. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 23–35, 2000.

[22] Raluca Vernic. Multivariate skew-normal distributions with applications in insurance. *Insurance: Mathematics and economics*, 38(2):413–426, 2006.

[23] Benoı^t Carmichael and Alain Coën. Asset pricing with skewed-normal return. *Finance Research Letters*, 10(2):50–57, 2013.

[24] Sylvia Frühwirth-Schnatter and Saumyadipta Pyne. Bayesian inference for finite mixtures of univariate and multivariate skew-normal and skew-t distributions. *Biostatistics*, 11(2):317–336, 2010.

[25] Kjartan Rimstad and Henning Omre. Skew-Gaussian random fields. *Spatial Statistics*, 10:43–62, 2014.

[26] Hao Zhang and Abdel El-Shaarawi. On spatial skew-Gaussian processes and applications. *Environmetrics: The official journal of the International Environmetrics Society*, 21(1):33–47, 2010.

[27] A. Azzalini and A. Capitanio. Statistical Applications of the Multivariate Skew Normal Distribution. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 61(3):579–602, September 1999.

[28] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4104–4113. IEEE Computer Society, 2016.

[29] Xingkui Wei, Yinda Zhang, Zhuwen Li, Yanwei Fu, and Xiangyang Xue. DeepSFM: Structure from Motion via Deep Bundle Adjustment. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, Lecture Notes in Computer Science, pages 230–247. Springer, 2020.

[30] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth Inference for Unstructured Multi-view Stereo. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VIII*, Lecture Notes in Computer Science, pages 785–801. Springer, 2018.

[31] Chenjie Cao, Xinlin Ren, and Yanwei Fu. MVSFormer: Multi-View Stereo by Learning Robust Image Features and Temperature-based Depth. *Trans. Mach. Learn. Res.*, 2022, 2022.

[32] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance Fields without Neural Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 5491–5500. IEEE, 2022.

[33] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 12479–12488. IEEE, 2023.

[34] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural Radiance Fields From One or Few Images. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 4578–4587. Computer Vision Foundation / IEEE, 2021.

[35] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas A. Funkhouser. IBRNet: Learning Multi-View Image-Based Rendering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 4690–4699. Computer Vision Foundation / IEEE, 2021.

[36] Hong-Xing Yu, Leonidas J. Guibas, and Jiajun Wu. Unsupervised Discovery of Object Radiance Fields. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[37] Di Qi, Tong Yang, and Xiangyu Zhang. Slot-guided Volumetric Object Radiance Fields. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[38] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume Rendering of Neural Implicit Surfaces. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 4805–4815, 2021.

[39] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 27171–27183, 2021.

[40] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 10318–10327. Computer Vision Foundation / IEEE, 2021.

[41] 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 Trans. Graph.*, 40(6):238:1–238:12, 2021.

[42] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-Splatting: Alias-Free 3D Gaussian Splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 19447–19456. IEEE, 2024.

[43] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-GS: Structured 3D Gaussians for View-Adaptive Rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 20654–20664. IEEE, 2024.

[44] Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-GS: Towards Consistent Real-time Rendering with LOD-Structured 3D Gaussians. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–15, 2025.

[45] Zheng Zhang, Wenbo Hu, Yixing Lao, Tong He, and Hengshuang Zhao. Pixel-GS: Density Control with Pixel-Aware Gradient for 3D Gaussian Splatting. In Ales Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XIX*, Lecture Notes in Computer Science, pages 326–342. Springer, 2024.

[46] Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. AbsGS: Recovering Fine Details in 3D Gaussian Splatting. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, page 1053–1061, New York, NY, USA, 2024. Association for Computing Machinery.

[47] Junjie Wang, Jiemin Fang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. GaussianEditor: Editing 3D Gaussians Delicately with Text Instructions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 20902–20911. IEEE, 2024.

[48] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. GaussianEditor: Swift and Controllable 3D Editing with Gaussian Splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 21476–21485. IEEE, 2024.

[49] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 20310–20320. IEEE, 2024.

[50] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[51] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[52] Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. GaussianDreamer: Fast Generation from Text to 3D Gaussians by Bridging 2D and 3D Diffusion Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 6796–6807. IEEE, 2024.

[53] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. LGM: Large Multi-view Gaussian Model for High-Resolution 3D Content Creation. In Ales Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part IV*, Lecture Notes in Computer Science, pages 1–18. Springer, 2024.

[54] Yinghao Xu, Zifan Shi, Yifan Wang, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. GRM: Large Gaussian Reconstruction Model for Efficient 3D Reconstruction and Generation. In Ales Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XV*, Lecture Notes in Computer Science, pages 1–20. Springer, 2024.

[55] Yanmin Wu, Jiarui Meng, Haijie Li, Chenming Wu, Yahao Shi, Xinhua Cheng, Chen Zhao, Haocheng Feng, Errui Ding, Jingdong Wang, and Jian Zhang. OpenGaussian: Towards

Point-Level 3D Gaussian-based Open Vocabulary Understanding. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[56] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. LangSplat: 3D Language Gaussian Splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 20051–20060. IEEE, 2024.

[57] Jun Guo, Xiaojian Ma, Yue Fan, Huaping Liu, and Qing Li. Semantic Gaussians: Open-Vocabulary Scene Understanding with 3D Gaussian Splatting. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2026.

[58] Jiazhong Cen, Jiemin Fang, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Segment Any 3D Gaussians. In Toby Walsh, Julie Shah, and Zico Kolter, editors, *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pages 1971–1979. AAAI Press, 2025.

[59] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian Grouping: Segment and Edit Anything in 3D Scenes. In Ales Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XXIX*, Lecture Notes in Computer Science, pages 162–179. Springer, 2024.

[60] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Trans. Graph.*, 36(4):78:1–78:13, 2017.

[61] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel J. Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.*, 37(6):257, 2018.

[62] Jiahui Zhang, Fangneng Zhan, Muyu Xu, Shijian Lu, and Eric P. Xing. FreGS: 3D Gaussian Splatting with Progressive Frequency Regularization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 21424–21433. IEEE, 2024.

[63] 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 Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Appendix

A Mathematical Proof

In the implementation of SNS, the kernel is defined using the unnormalized probability density function (PDF) of the Skew-Normal distribution. Although this differs from the standard Skew-Normal (SN) distribution by a normalization coefficient, they share identical properties regarding affine transformations and geometric characteristics. For clarity and convenience, the following derivations are conducted within the framework of the standard SN distribution.

A.1 Closure under Affine Transformations

Let $X \sim SN_d(\mu, \Omega, \alpha)$ be a d-dimensional Skew-Normal random variable. Its stochastic representation [27] is given by:

$$X = \mu + \omega Z \mid U > 0, \quad \begin{pmatrix} U \\ Z \end{pmatrix} \sim N_{1+d}\left(\mathbf{0}, \begin{pmatrix} 1 & \delta^\top \\ \delta & \bar{\Omega} \end{pmatrix}\right), \tag{12}$$

where $\omega = (\Omega \odot \mathbf{I}_d)^{1/2}$, $\bar{\Omega} = \omega^{-1}\Omega\omega^{-1}$, and $\delta = \frac{\bar{\Omega}\alpha}{\sqrt{1+\alpha^\top\bar{\Omega}\alpha}}$.

Consider an affine transformation $Y = TX + b$, where $T \in \mathbb{R}^{h\times d}$, $\text{rank}(T) = h \leq d$ and $b \in \mathbb{R}^h$. Substituting the representation of X yields: $Y = (T\mu + b) + T\omega Z \mid U > 0$.

To show that Y is also Skew-Normal, we express it in the standard stochastic representation. Let $\mu_Y = T\mu + b$ and $W = T\omega Z$. Since $(U, Z^\top)^\top$ is multivariate normal, the joint distribution of $(U, W^\top)^\top$ remains normal with zero mean. The variance of W and the covariance between U and W are given by:

$$\begin{aligned} \text{Var}(W) &= \Omega_Y = T\omega\bar{\Omega}\omega T^\top = T\Omega T^\top, \\ \text{Cov}(U, W) &= \text{Cov}(U, Z)\omega T^\top = \delta^\top\omega T^\top. \end{aligned} \tag{13}$$

Defining $\omega_Y = (\Omega_Y \odot \mathbf{I}_h)^{1/2}$ and $Z_Y = \omega_Y^{-1}W$, we can rewrite $Y = \mu_Y + \omega_Y Z_Y \mid U > 0$, while $\text{Var}(Z_Y) = \omega_Y^{-1}\Omega_Y\omega_Y^{-1} = \bar{\Omega}_Y$, and $\text{Cov}(U, Z_Y) = \text{Cov}(U, W)\omega_Y^{-1} = \delta^\top\omega T^\top\omega_Y^{-1}$. Letting $\delta_Y = \omega_Y^{-1}T\omega\delta$, we finally obtain the required block-covariance structure:

$$Y = \mu_Y + \omega_Y Z_Y \mid U > 0, \quad \begin{pmatrix} U \\ Z_Y \end{pmatrix} \sim N_{1+h}\left(\mathbf{0}, \begin{pmatrix} 1 & \delta_Y^\top \\ \delta_Y & \bar{\Omega}_Y \end{pmatrix}\right). \tag{14}$$

Since the conditioning event $U > 0$ is unchanged and $(U, Z_Y^\top)^\top$ satisfies the standard joint normal structure, we conclude that Y follows an h-dimensional SND.

A.2 2D Kernel Parameters

Let $X \sim \text{SN}_3(\mu, \Omega, \alpha)$ represent the 3D Skew-Normal kernel, and consider a linear projection: $Y = AX$, where $A \in \mathbb{R}^{2\times 3}$. Building upon the property derived in A.1, we can simply write $Y \sim SN_2(\mu_Y, \Omega_Y, \alpha_Y)$, with $\mu_Y = A\mu$, $\Omega_Y = A\Omega A^\top$, and $\delta_Y = \omega_Y^{-1}A\omega\delta$.

Recall that $\delta_Y = \frac{\bar{\Omega}_Y\alpha_Y}{\sqrt{1+\alpha_Y^\top\bar{\Omega}_Y\alpha_Y}}$ and $1 - \delta_Y^\top\bar{\Omega}_Y^{-1}\delta_Y = \frac{1}{1+\alpha_Y^\top\bar{\Omega}_Y\alpha_Y}$. We can write:

$$\alpha_Y = \sqrt{1 + \alpha_Y^\top\bar{\Omega}_Y\alpha_Y} \cdot \bar{\Omega}_Y^{-1}\delta_Y = \frac{\bar{\Omega}_Y^{-1}\delta_Y}{\sqrt{1 - \delta_Y^\top\bar{\Omega}_Y^{-1}\delta_Y}} = \frac{\omega_Y\Omega_Y^{-1}A\omega\delta}{\sqrt{1 - (A\omega\delta)^\top\Omega_Y^{-1}A\omega\delta}}, \tag{15}$$

where $\omega_Y = (\Omega_Y \odot \mathbf{I}_2)^{1/2}$, $\omega = (\Omega \odot \mathbf{I}_3)^{1/2}$, and $\delta = \frac{\bar{\Omega}\alpha}{\sqrt{1+\alpha^\top\bar{\Omega}\alpha}}$.

In our formulation, α is explicitly parameterized as $\alpha = \omega RS^{-1}k$, where $\Omega = RSS^\top R^\top$. Substituting this structural definition into the equations above simplifies the expressions:

$$\begin{aligned}\omega\delta &= \frac{\omega\bar{\Omega}\omega RS^{-1}k}{\sqrt{1+k^\top S^{-1}R^\top\omega\bar{\Omega}\omega RS^{-1}k}} = \frac{\Omega RS^{-1}k}{\sqrt{1+k^\top S^{-1}R^\top\Omega RS^{-1}k}} \\ &= \frac{RSS^\top R^\top RS^{-1}k}{\sqrt{1+k^\top S^{-1}R^\top RSS^\top R^\top RS^{-1}k}} = \frac{RSk}{\sqrt{1+k^\top k}}.\end{aligned} \tag{16}$$

Applying the transformation Eq.(16) to Eq.(15) gives:

$$\alpha_Y = \frac{\omega_Y\Omega_Y^{-1}ARSk}{\sqrt{1+k^\top k-(ARSk)^\top(\Omega_Y)^{-1}ARSk}}. \tag{17}$$

A.3 The Mean of the Skew-Normal Distribution

Unlike the multivariate normal distribution, the location parameter μ of $X \sim SN_d(\mu, \Omega, \alpha)$ does not equal its exact mean. The true mean, $\mathbb{E}[X]$, can be analytically derived using the stochastic representation introduced in Eq.(12). Taking the expectation of $X = \mu + \omega Z \mid U > 0$, we have:

$$\begin{aligned}\mathbb{E}[X] &= \mu + \omega\mathbb{E}[Z \mid U > 0] \\ &= \mu + \omega\mathbb{E}[\mathbb{E}[Z \mid U] \mid U > 0] \\ &= \mu + \omega\left([\mathbb{E}[Z - U\delta] + \mathbb{E}[U\delta \mid U > 0]\right) \\ &= \mu + \mathbb{E}[U \mid U > 0] \cdot \omega\delta \\ &= \mu + \sqrt{\frac{2}{\pi}} \cdot \omega\delta.\end{aligned} \tag{18}$$

If $\alpha = \omega RS^{-1}k$, associated with Eq.(16), we have:

$$\mathbb{E}[X] = \mu + \sqrt{\frac{2}{\pi}} \cdot \frac{RSk}{\sqrt{1+k^\top k}} \tag{19}$$

B Backward Pass

The learnable parameters for each component in SNS include the location μ, scale matrix $\Omega(i.e.\ S$ and $R)$, color c, opacity o, skew k, and the key gradients are $\frac{\partial L}{\partial \mu}$, $\frac{\partial L}{\partial S}$, $\frac{\partial L}{\partial R}$, $\frac{\partial L}{\partial c}$, $\frac{\partial L}{\partial o}$, $\frac{\partial L}{\partial k}$, where L is the loss. L is calculated between the ground-truth pixel colors and rendered colors. The rendered pixel colors are:

$$C(u) = \sum_{i=1}^{N} c_i a_i(u) \prod_{j=1}^{i-1} \left(1 - a_j(u)\right), \tag{20}$$

where $a(u) = o \cdot SN^{2D}(u)$ and u denotes pixel coordinate in 2D images.

Previous methods, such as 3DGS and SSS, compute the gradients with respect to the shape parameters (R, S) by first calculating $\frac{\partial L}{\partial(\Omega^{-1})}$ and subsequently backpropagating to obtain $\frac{\partial L}{\partial R}$ and $\frac{\partial L}{\partial S}$. However, recalling the definitions in Eq. (5) and Eq. (6), computing $\frac{\partial L}{\partial \Omega}$ directly proves cumbersome. The parameters (R, S) couple intricately within both $ARSk$ and $(\Omega^{2D})^{-1}$, leading to high computational complexity. To mitigate this, we propose an alternative differentiation scheme. By introducing an intermediate variable $Q = ARS$, we reformulate Eq. (5) and Eq. (6) as follows:

$$SN^{2D}(u) = G_2(u - \mu^{2D}; QQ^\top)\, \Phi\left((m^{2D})^\top(u - \mu^{2D})\right), \tag{21}$$

where

$$m^{2D} = \frac{(QQ^\top)^{-1}Qk}{\sqrt{1+k^\top k - k^\top Q^\top(QQ^\top)^{-1}Qk}}.$$

Applying the chain rule, the gradients are given by:

$$\begin{aligned}
\frac{\partial L}{\partial c} &= \frac{\partial L}{\partial C} \cdot \frac{\partial C}{\partial c}, \\
\frac{\partial L}{\partial o} &= \frac{\partial L}{\partial C} \cdot \frac{\partial C}{\partial a} \cdot \frac{\partial a}{\partial o}, \\
\frac{\partial L}{\partial \mu} &= \frac{\partial L}{\partial C} \cdot \frac{\partial C}{\partial a} \cdot \frac{\partial a}{\partial SN^{2D}} \cdot \frac{\partial SN^{2D}}{\partial \mu^{2D}} \cdot \frac{\partial \mu^{2D}}{\partial \mu}, \\
\frac{\partial L}{\partial k} &= \frac{\partial L}{\partial C} \cdot \frac{\partial C}{\partial a} \cdot \frac{\partial a}{\partial SN^{2D}} \cdot \frac{\partial SN^{2D}}{\partial k}, \\
\frac{\partial L}{\partial R} &= \frac{\partial L}{\partial C} \cdot \frac{\partial C}{\partial a} \cdot \frac{\partial a}{\partial SN^{2D}} \cdot \frac{\partial SN^{2D}}{\partial Q} \cdot \frac{\partial Q}{\partial R}, \\
\frac{\partial L}{\partial S} &= \frac{\partial L}{\partial C} \cdot \frac{\partial C}{\partial a} \cdot \frac{\partial a}{\partial SN^{2D}} \cdot \frac{\partial SN^{2D}}{\partial Q} \cdot \frac{\partial Q}{\partial S},
\end{aligned} \tag{22}$$

where C and a are defined in Eq. (20). Compared to the standard gradient computation in 3DGS, our formulation solely requires deriving the partial derivatives of SN^{2D} with respect to μ^{2D}, k, and Q; the gradients for all other standard components remain unchanged and are therefore omitted. The final analytical expressions for these specific derivatives are presented in Eq. (27), Eq. (30), and Eq. (40), respectively. For notational simplicity, we denote $m = m^{2D}$, $\Delta = u - \mu^{2D}$, and $D = \sqrt{1 + k^\top k - k^\top Q^\top (\Omega^{2D})^{-1} Q k}$. Also, we define the following auxiliary variables:

$$\begin{aligned}
\boldsymbol{c}_\Delta &= \Phi \cdot G_2 \cdot (\Omega^{2D})^{-1}\Delta, \\
\boldsymbol{M}_\Delta &= \Phi \cdot G_2 \cdot \left((\Omega^{2D})^{-1}\Delta\right)\left((\Omega^{2D})^{-1}\Delta\right)^\top, \\
s_\Delta &= \frac{G_2}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(m^\top \Delta)^2\right), \\
\boldsymbol{p}_\Delta &= \frac{G_2}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(m^\top \Delta)^2\right) \cdot (\Omega^{2D})^{-1}\Delta,
\end{aligned} \tag{23}$$

where $s_\Delta \in \mathbb{R}$, $\boldsymbol{c}_\Delta, \boldsymbol{p}_\Delta \in \mathbb{R}^2$ and $\boldsymbol{M}_\Delta \in \mathbb{R}^{2\times 2}$.

Gradient with respect to μ^{2D}.

First, we derive the gradient with respect to the mean vector μ^{2D}. Applying the product rule, we obtain:

$$\frac{\partial SN^{2D}}{\partial \mu^{2D}} = \Phi \cdot \frac{\partial G_2}{\partial \mu^{2D}} + G_2 \cdot \frac{\partial \Phi}{\partial \mu^{2D}}. \tag{24}$$

Given $\Delta = u - \mu^{2D}$, the inner derivative is $\frac{\partial \Delta}{\partial \mu^{2D}} = -I$. So the derivative of the Gaussian component G_2 is:

$$\frac{\partial G_2}{\partial \mu^{2D}} = G_2 (\Omega^{2D})^{-1}(u - \mu^{2D}) = G_2 (\Omega^{2D})^{-1}\Delta. \tag{25}$$

For the cumulative distribution function (CDF) component $\Phi(m^\top \Delta)$, the chain rule yields:

$$\frac{\partial \Phi(m^\top \Delta)}{\partial \mu^{2D}} = -\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(m^\top \Delta)^2\right) m. \tag{26}$$

Substituting these partial derivatives back into the product rule and utilizing the definitions in Eq. (23), we arrive at the intermediate expression:

$$\begin{aligned}
\frac{\partial SN^{2D}}{\partial \mu^{2D}} &= \Phi \cdot G_2 (\Omega^{2D})^{-1}\Delta - \frac{G_2}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(m^\top \Delta)^2\right) m \\
&= \boldsymbol{c}_\Delta - s_\Delta \cdot m.
\end{aligned} \tag{27}$$

Gradient with respect to k.

Next, we derive the gradient with respect to k:

$$\begin{aligned}\frac{\partial SN^{2D}}{\partial k} &= G_2 \cdot \frac{\partial \Phi}{\partial (m^\top \Delta)} \cdot \frac{\partial (m^\top \Delta)}{\partial k} \\ &= \frac{G_2}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(m^\top \Delta)^2\right) \cdot \frac{\partial (m^\top \Delta)}{\partial k},\end{aligned} \tag{28}$$

where

$$\begin{aligned}\frac{\partial (m^\top \Delta)}{\partial k} =& \frac{1}{D} \cdot \frac{\partial \left(\Delta^\top (\Omega^{2D})^{-1} Q k\right)}{\partial k} \\ & - \frac{\Delta^\top (\Omega^{2D})^{-1} Q k}{2D^3} \cdot \frac{\partial \left(k^\top k - k^\top Q^\top (\Omega^{2D})^{-1} Q k\right)}{\partial k} \\ =& \frac{Q^\top (\Omega^{2D})^{-1} \Delta}{D} - \frac{\Delta^\top (\Omega^{2D})^{-1} Q k}{D^3} \cdot \left(k - Q^\top (\Omega^{2D})^{-1} Q k\right) \\ =& \frac{Q^\top (\Omega^{2D})^{-1} \Delta}{D} + \frac{(\Delta^\top m) \cdot Q^\top m}{D} - \frac{(\Delta^\top m) \cdot k}{D^2} \\ =& \frac{1}{D} Q^\top \left((\Omega^{2D})^{-1} \Delta + (\Delta^\top m) \cdot m\right) - \frac{1}{D^2} \cdot (\Delta^\top m) \cdot k.\end{aligned} \tag{29}$$

Combining Eq. (23), Eq. (28), and Eq. (29), we obtain:

$$\frac{\partial SN^{2D}}{\partial k} = \frac{1}{D} Q^\top \left(\boldsymbol{p}_\Delta + (\boldsymbol{p}_\Delta^\top \Omega^{2D} m) \cdot m\right) - \frac{1}{D^2} \cdot (\boldsymbol{p}_\Delta^\top \Omega^{2D} m) \cdot k. \tag{30}$$

Gradient with respect to Q.

To compute $\frac{\partial SN^{2D}}{\partial Q}$, we apply the product rule to the respective terms:

$$\frac{\partial SN^{2D}}{\partial Q} = G_2 \cdot \frac{\partial \Phi}{\partial (m^\top \Delta)} \cdot \frac{\partial (m^\top \Delta)}{\partial Q} + \Phi \cdot \frac{\partial G_2}{\partial Q}. \tag{31}$$

First, we calculate the gradient of the 2D Gaussian G_2 with respect to Q. The differential of G_2 can be expressed using the trace operator $\text{Tr}(\cdot)$:

$$\begin{aligned}dG_2 &= G_2 \cdot \left(-\frac{1}{2} \Delta^\top d(\Omega^{2D})^{-1} \Delta\right) \\ &= \frac{1}{2} G_2 \text{Tr}\left(\Delta^\top (\Omega^{2D})^{-1} (Q dQ^\top + dQ Q^\top)(\Omega^{2D})^{-1} \Delta\right) \\ &= \text{Tr}\left(G_2 \cdot Q^\top (\Omega^{2D})^{-1} \Delta \Delta^\top (\Omega^{2D})^{-1} dQ\right).\end{aligned} \tag{32}$$

By the matrix calculus identity where the differential $df = \text{Tr}(P^\top dQ)$ implies the gradient $\frac{\partial f}{\partial Q} = P$, we extract the derivative with respect to Q:

$$\frac{\partial G_2}{\partial Q} = G_2 \left((\Omega^{2D})^{-1} \Delta \Delta^\top (\Omega^{2D})^{-1}\right) Q. \tag{33}$$

Next, we compute $\frac{\partial (m^\top \Delta)}{\partial Q}$. Let $N = \Delta^\top (\Omega^{2D})^{-1} Q k$, which implies $m^\top \Delta = \frac{N}{D}$. By the quotient rule, its differential is:

$$d(m^\top \Delta) = \frac{1}{D} dN - \frac{N}{D^2} dD = \frac{1}{D} dN - \frac{m^\top \Delta}{D} dD. \tag{34}$$

For dN, evaluating the differential and substituting $(\Omega^{2D})^{-1} Q k = Dm$, we have:

$$\begin{aligned}dN &= \Delta^\top d\left((\Omega^{2D})^{-1}\right) Q k + \Delta^\top (\Omega^{2D})^{-1} dQ\, k \\ &= -\Delta^\top (\Omega^{2D})^{-1} (dQ\, Q^\top + Q\, dQ^\top)(\Omega^{2D})^{-1} Q k + \Delta^\top (\Omega^{2D})^{-1} dQ\, k \\ &= \text{Tr}\left(k \Delta^\top (\Omega^{2D})^{-1} dQ - Q^\top D m \Delta^\top (\Omega^{2D})^{-1} dQ - D m \Delta^\top (\Omega^{2D})^{-1} Q dQ^\top\right) \\ &= \text{Tr}\left(\left[(\Omega^{2D})^{-1} \Delta k^\top - D(\Omega^{2D})^{-1} \Delta m^\top Q - D m \Delta^\top (\Omega^{2D})^{-1} Q\right]^\top dQ\right).\end{aligned} \tag{35}$$

This establishes the gradient of N:

$$\frac{\partial N}{\partial Q} = (\Omega^{2D})^{-1}\Delta k^\top - D\left((\Omega^{2D})^{-1}\Delta m^\top + m\Delta^\top(\Omega^{2D})^{-1}\right)Q. \tag{36}$$

Similarly, we derive variable D's differential:

$$\begin{aligned} dD &= -\frac{1}{2D}\cdot d\left(k^\top Q^\top(\Omega^{2D})^{-1}Qk\right) \\ &= -\frac{1}{D}\cdot\left(k^\top Q^\top(\Omega^{2D})^{-1}dQ\,k - k^\top Q^\top(\Omega^{2D})^{-1}dQ\,Q^\top(\Omega^{2D})^{-1}Qk\right) \\ &= -\frac{1}{D}\cdot\left(Dm^\top dQ\,k - D^2 m^\top dQ\,Q^\top m\right) \\ &= \mathrm{Tr}\left(\left(-mk^\top + D\cdot mm^\top Q\right)^\top dQ\right), \end{aligned} \tag{37}$$

which yields:

$$\frac{\partial D}{\partial Q} = -mk^\top + D\cdot mm^\top Q. \tag{38}$$

Combining Eq. (34), Eq. (36), and Eq. (38), we assemble $\frac{\partial(m^\top\Delta)}{\partial Q}$:

$$\begin{aligned} \frac{\partial(m^\top\Delta)}{\partial Q} =& \frac{1}{D}\left((\Omega^{2D})^{-1}\Delta + (m^\top\Delta)m\right)k^\top \\ &- \left((\Omega^{2D})^{-1}\Delta m^\top + m\Delta^\top(\Omega^{2D})^{-1} + (m^\top\Delta)mm^\top\right)Q. \end{aligned} \tag{39}$$

Finally, recalling that $G_2\cdot\frac{\partial\Phi}{\partial(m^\top\Delta)} = \frac{G_2}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}(m^\top\Delta)^2\right)$, we multiply Eq. (39) by this term. Substituting the previously defined variables $\boldsymbol{p}_\Delta$ and M_Δ, the full analytical gradient $\frac{\partial SN^{2D}}{\partial Q}$ is derived as:

$$\begin{aligned} \frac{\partial SN^{2D}}{\partial Q} =& \left(\boldsymbol{M}_\Delta - \boldsymbol{p}_\Delta m^\top - m\boldsymbol{p}_\Delta^\top - (\boldsymbol{p}_\Delta^\top\Omega^{2D}m)\cdot mm^\top\right)Q \\ &+ \frac{1}{D}\left(\boldsymbol{p}_\Delta + (\boldsymbol{p}_\Delta^\top\Omega^{2D}m)\cdot m\right)k^\top. \end{aligned} \tag{40}$$

Following the standard 3DGS architecture, the backward pass consists of rendering and preprocessing stages. In standard 3DGS, the rendering stage first computes gradients with respect to 2D image-space variables (e.g., $\frac{\partial L}{\partial\mu^{2D}}$ and $\frac{\partial L}{\partial(\Omega^{2D})^{-1}}$), and the preprocessing stage uses these intermediate gradients to compute the gradients for 3D learnable parameters.

By contrast, the SNS rendering stage directly evaluates the shared auxiliary terms ($\boldsymbol{c}_\Delta, \boldsymbol{M}_\Delta, s_\Delta, \boldsymbol{p}_\Delta$), which are intermediate algebraic quantities rather than derivatives. Next, the preprocessing stage leverages these precomputed terms to efficiently evaluate the exact gradients of the 3D parameters using our derived closed-form expressions (Eq. (27), Eq. (30), and Eq. (40)). By algebraically restructuring the complex gradient derivation into clear and concise computations, this formulation maximizes variable reuse and ensures a smooth and efficient computational flow.

C Detailed results

In this section, we present the detailed results. Specifically, Table 4 reports the complete PSNR, SSIM, and LPIPS metrics for our full SNS model and its ablations across all evaluated scenes.

D Alternating Optimization Strategy

Jointly optimizing the base geometric attributes (rotation and scaling) with the skewness parameters can lead to optimization instability due to their highly coupled influence on the primitive's spatial footprint. To mitigate this, we introduce an Alternating Optimization strategy inspired by Block Coordinate Descent (BCD).

Table 4: **Per-scene ablation details.** This table presents the performance of our Full SNS compared with its ablated versions. The Full SNS is composed of a baseline combined with k-decomposition and Alternating Optimization. We report PSNR, SSIM, and LPIPS for each scene and the average per dataset.

Dataset	Scene	Vanilla SNS PSNR↑ / SSIM↑ / LPIPS↓	k-decomposition PSNR↑ / SSIM↑ / LPIPS↓	Alternating Optimization PSNR↑ / SSIM↑ / LPIPS↓	Full SNS PSNR↑ / SSIM↑ / LPIPS↓
Mip-NeRF 360	Counter	30.1129 / 0.9270 / 0.1534	30.2460 / 0.9283 / 0.1531	30.1116 / 0.9277 / 0.1540	30.2749 / 0.9282 / 0.1537
	Bicycle	25.7461 / 0.8004 / 0.1719	25.8033 / 0.8008 / 0.1726	25.8162 / 0.8011 / 0.1724	25.8510 / 0.8024 / 0.1722
	Stump	27.2857 / 0.8144 / 0.1739	27.3576 / 0.8151 / 0.1744	27.3833 / 0.8144 / 0.1756	27.3130 / 0.8149 / 0.1755
	Garden	28.1096 / 0.8828 / 0.0909	28.1546 / 0.8832 / 0.0905	28.1348 / 0.8830 / 0.0910	28.2243 / 0.8837 / 0.0908
	Bonsai	33.6172 / 0.9551 / 0.1514	33.7117 / 0.9563 / 0.1505	33.7363 / 0.9562 / 0.1509	33.7026 / 0.9568 / 0.1514
	Room	33.0042 / 0.9392 / 0.1662	32.9194 / 0.9397 / 0.1660	33.0393 / 0.9401 / 0.1658	33.0163 / 0.9400 / 0.1665
	Kitchen	32.5742 / 0.9402 / 0.1022	32.6499 / 0.9415 / 0.1020	32.7514 / 0.9419 / 0.1017	32.8082 / 0.9420 / 0.1016
	Average	30.0643 / 0.8942 / 0.1443	30.1204 / 0.8950 / 0.1442	30.1390 / 0.8949 / 0.1445	30.1700 / 0.8954 / 0.1445
Deep Blending	Playroom	30.4695 / 0.9099 / 0.2446	30.3995 / 0.9144 / 0.2425	30.6745 / 0.9153 / 0.2410	30.7130 / 0.9148 / 0.2425
	DrJohnson	29.8467 / 0.9095 / 0.2450	29.8375 / 0.9091 / 0.2456	29.9192 / 0.9091 / 0.2461	29.8938 / 0.9089 / 0.2462
	Average	30.1581 / 0.9097 / 0.2448	30.1185 / 0.9118 / 0.2441	30.2969 / 0.9122 / 0.2436	30.3034 / 0.9119 / 0.2444
Tanks & Temples	Train	23.5475 / 0.8548 / 0.1604	23.5138 / 0.8548 / 0.1614	23.5132 / 0.8564 / 0.1606	23.6065 / 0.8561 / 0.1615
	Truck	26.5301 / 0.8981 / 0.1044	26.5041 / 0.8987 / 0.1024	26.5523 / 0.8987 / 0.1037	26.5613 / 0.8989 / 0.1035
	Average	25.0388 / 0.8765 / 0.1324	25.0090 / 0.8768 / 0.1319	25.0328 / 0.8776 / 0.1322	25.0839 / 0.8775 / 0.1325

Algorithm 1 Alternating Optimization for Primitive Shape

Require: Current iteration step t, warm-up threshold T_{start}, cycle length C, base phase duration C_{base}.
Require: Base shape parameters Θ_{base}, skewness parameters Θ_{skew}.
Compute loss $\mathcal{L}$ and initial gradients for all parameters
if $t > T_{start}$ **then** ▷ Begin Alternating Optimization
  $step_{cycle} \leftarrow t \pmod{C}$
  if $step_{cycle} < C_{base}$ **then**
    Freeze Θ_{skew} ▷ Zero gradients & LR for skewness
    Unfreeze Θ_{base}
  else
    Freeze Θ_{base} ▷ Zero gradients & LR for base shape
    Unfreeze Θ_{skew}
  end if
else
  Unfreeze all Θ_{base} and Θ_{skew} ▷ Joint optimization
end if
Apply optimizer step (e.g., Adam/SGHMC) to update unfrozen parameters

As outlined in Algorithm 1, we partition the shape-related parameters into two disjoint groups: the base shape parameters $\Theta_{base} = \{R, S\}$ and the skewness parameters $\Theta_{skew} = \{d_k, m_k\}$. During the initial warm-up phase (before iteration T_{start}), all parameters are optimized jointly to establish a reasonable geometry. Once the threshold T_{start} is reached, we interleave the optimization of Θ_{base} and Θ_{skew} within a fixed cycle length C. In each cycle, we freeze the skewness to refine the base shape for C_{base} steps, and subsequently freeze the base shape to optimize the skewness for the remaining steps. This decoupled optimization stabilizes the gradient descent and prevents degenerate primitive shapes.

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: We summarize our contribution accurately in our abstract and introduction.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Please see Section 6.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: Our paper does not present formal theoretical results. The mathematical derivations, provided in the appendix, serve to derive the computational formulas for our proposed method.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The method and its derivations are described in the paper and appendix. We have publicly released our code on GitHub.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We have publicly released our code on GitHub. Our method is evaluated on widely used public datasets.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The experimental setup is described in Section 4. More details are provided in the released code repository.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: We have not provided the repeated experiments for the error bar.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [No]

   Justification: We do not provide the memory and time of execution.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Our research conforms to the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: This paper has no clear societal impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: This paper poses no such risk.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer:[Yes]

    Justification: We properly cited the previous works.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.