Warp-2DGS: Learnable Warped Kernels for 2D Gaussian Splatting

Huibin Li **Chul Min Yeum**
CVISS Lab
University of Waterloo
{h844li, cmyeum}@uwaterloo.ca

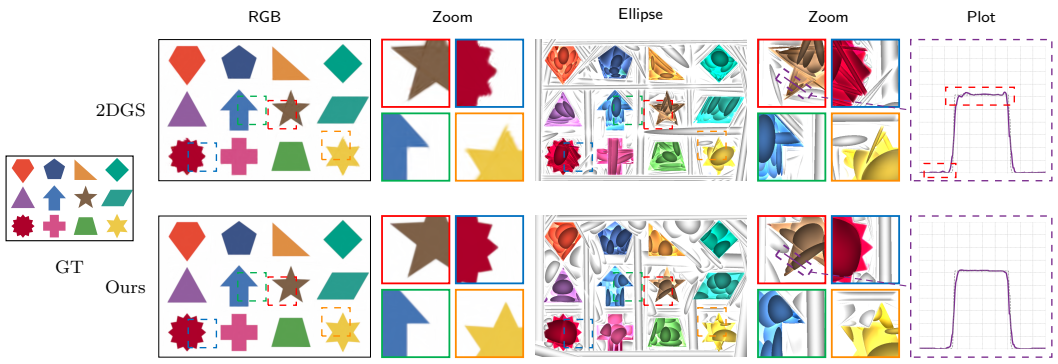


Figure 1: **Warped kernels suppress Gibbs.** Comparison between 2DGS (top) and Warp-2DGS (bottom) on a synthetic 2D scene. Dashed insets zoom into regions with sharp boundary features, where 2DGS suffers from Gibbs-like ringing artifacts caused by superimposing many Gaussian kernels to approximate discontinuities. Right (Plot): inverted gray value along the purple scanline. Our warped kernels can adopt a step-like edge region that suppresses such oscillatory overshoots.

Abstract

Building upon the real-time rendering capability of 3D Gaussian Splatting (3DGS), 2D Gaussian Splatting (2DGS) achieves improved surface reconstruction and geometric fidelity for novel view synthesis through explicit planar Gaussian primitives. However, the smooth falloff inherent to Gaussian kernels fundamentally limits the reconstruction of high-frequency shape signals, causing over-smoothed edges and loss of fine geometric detail. Prior works have explored alternative kernel functions to address this limitation, yet through systematic evaluation across signal fitting tasks, we find that all existing kernels suffer from the Gibbs phenomenon — producing oscillatory artifacts near sharp discontinuities. This observation motivates a fundamentally different strategy: rather than replacing the kernel, we make it deformable. This study presents *Warp-2DGS*, which augments each Gaussian surfel with a learnable, monotonic coordinate remapping in its local tangent plane. With only one free control point per axis (two parameters per axis, four per primitive), the remapping enables asymmetric radial profiles, allowing each primitive to reliably capture sharp edges and irregular structures that symmetric kernels inherently cannot represent. The deformation operates on the 2D tangent plane of 2DGS, preserving the differentiable rasterization pipeline with moderate training overhead, and is optimized end-to-end through rendering loss alone. Experiments on MipNeRF 360 and Tanks & Temples demonstrate that our method effectively suppresses Gibbs artifacts at sharp boundaries and achieves the best PSNR among

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

2DGS-based approaches, with consistent gains over 2DGS in PSNR, SSIM, and LPIPS.

1 Introduction

Neural radiance fields have significantly advanced photorealistic novel view synthesis by encoding scenes as implicit representations. While 3D Gaussian Splatting (3DGS) introduced an explicit alternative using anisotropic 3D Gaussian primitives for real-time rendering, its volumetric nature often leads to noisy geometry and multi-surface artifacts. 2D Gaussian Splatting (2DGS) addresses these limitations by collapsing the primitives to planar, surfel-like 2D Gaussians, yielding more accurate surface reconstruction while preserving real-time performance. Its improved geometric fidelity and inherent surface alignment have driven growing adoption across applications that demand both rendering quality and reliable geometry [35, 32, 29, 25, 8, 5, 31].

Despite these advantages, the standard Gaussian kernel inherently lacks the ability to represent high-frequency shape signals. This limitation leads to over-smoothed reconstructions, particularly along sharp edges and fine geometric details. As illustrated in Figure 1, existing methods exhibit visible artifacts in these regions, where the smooth Gaussian falloff fails to capture abrupt signal transitions.

Recent efforts have attempted to address this issue by replacing the Gaussian kernel with alternative primitives [10, 22, 23, 36, 26, 3, 21, 17, 15]. However, through systematic evaluation of a variety of existing kernel functions in signal fitting tasks, we observe that the problem remains challenging to fully resolve. In particular, all tested kernels give rise to the Gibbs [9] phenomenon when approximating sharp boundaries, producing characteristic oscillatory artifacts near discontinuities. This finding suggests that a fundamentally different kernel design is required — one that can faithfully represent sharp edge signals while remaining smooth enough to be differentiable and trainable. Furthermore, since shape signals in real scenes are spatially non-uniform, a fully symmetric kernel is inherently limited in its expressiveness.

To address these challenges, we propose a warped Gaussian kernel that provides asymmetric and adaptive shape representation while preserving differentiability for end-to-end optimization. The deformation is realized by composing the Gaussian with a learnable radial warping function defined by a set of free control points on the local tangent plane introduced by 2DGS, allowing each primitive to reshape its radial profile into arbitrary monotonically decreasing forms beyond the fixed exponential falloff. Operating directly on the 2D tangent plane avoids deforming a 3D volumetric kernel and projecting it onto the image plane, which would otherwise require costly numerical integration or approximation. Our contributions are summarized as follows:

- We introduce a warped Gaussian kernel that combines asymmetric learning capacity with differentiability, enabling sharp boundary representation that symmetric smooth kernels inherently cannot achieve.
- We demonstrate that our approach effectively mitigates the Gibbs phenomenon at sharp edges, a fundamental limitation inherent in existing kernel alternatives.
- Our method achieves the best PSNR among 2DGS-based approaches on standard benchmarks.

2 Related Work

The standard Gaussian kernel enables efficient differentiable rasterization but its symmetric radial falloff struggles to represent sharp geometric features such as edges, occlusion boundaries, and thin structures. Real-world scenes often contain spatially asymmetric signals that a fixed symmetric kernel cannot faithfully capture with a single primitive. A growing body of work has sought to address this limitation by modifying the kernel function or replacing the primitive altogether [10, 22, 23, 36, 26, 3, 21, 17, 15].

Kernel function. GES [10] makes the exponent of the Gaussian function learnable, allowing each primitive to control its falloff sharpness. BetaGS [22] replaces the Gaussian with a Beta distribution kernel that offers bounded support, adaptive frequency control, and asymmetric shape modeling, and its successor, Universal Beta Splatting [23], extends this to N-dimensional anisotropic settings.

StudentGS [36] adopts the Student-t distribution for heavier tails. DARB Splatting [26] systematically compares a family of decaying anisotropic radial basis functions, including half-cosine, raised-cosine, and multiquadric kernels. 3DLS [3] replaces the exponential with a bounded linear kernel to sharpen primitive boundaries and reduce floaters. 3D-Half GS [21] splits each Gaussian into two halves with independent opacities, enabling one-sided sharp edges. NegGS [17] introduces negative-amplitude Gaussians to counteract the long-tail effect through destructive interference. DRK [15] enriches 2D Gaussian surfels with learnable radial basis functions defined by multiple free control points to represent arbitrary closed shapes. DRK changes the footprint angularly, whereas our warp changes the falloff along each tangent axis. The two act in complementary directions and could, in principle, be combined. However, despite their varied radial falloffs and asymmetric shapes, these kernels still exhibit the Gibbs phenomenon when fitting sharp edges or high-frequency content (see Section 3.2).

Gaussian primitive with alternative geometric representations. 3DCS [12] uses a smooth convex polyhedron to better resolve planar surfaces and edges. Triangle Splatting [13] optimizes explicit triangles, bridging the gap to traditional mesh-based rendering pipelines. BG-Triangle [30] combines Bézier triangle surfaces with pixel-aligned Gaussian sub-primitives for vectorized scene representation with sharp boundary preservation. QGS [34] extends the planar surfel to a quadratic surface model, enabling each primitive to capture local curvature for improved geometric fitting. These approaches change the geometry of the primitive itself and are thus complementary to kernel-level modifications. They do not address the limited expressiveness of the falloff profile within a planar primitive, which is why our main comparison focuses on methods that retain a planar primitive.

Our approach is orthogonal to these methods. Rather than modifying the kernel function or replacing the primitive, we introduce a learnable coordinate remapping in the local $u - v$ tangent plane of each 2D Gaussian surfel. The Gaussian kernel itself remains unchanged; instead, the input coordinates are passed through a monotonic function before evaluation, enabling asymmetric radial falloff with minimal additional parameters. Since our method operates on the input coordinates through a learnable radial mapping rather than replacing the kernel function itself, it offers a more general formulation that can recover similar effective profiles under suitable parameterizations.

3 Method

3.1 Preliminaries: 2D Gaussian Splatting

Gaussian Splatting methods represent scenes using collections of Gaussian primitives, typically initialized from point clouds obtained via Structure-from-Motion (SfM) [28] or LiDAR. While 3DGS [18] uses anisotropic 3D ellipsoids, its volumetric nature often leads to view-inconsistent geometry. 2DGS [14] addresses this by reformulating each primitive as a 2D oriented planar disk embedded in 3D space, enabling more accurate surface reconstruction. Each 2D Gaussian is parameterized by a set of learnable attributes: center $\boldsymbol{\mu} \in \mathbb{R}^3$, two principal tangent vectors $\mathbf{t}_u, \mathbf{t}_v \in \mathbb{R}^3$ spanning the local surface plane, corresponding scaling parameters $s_u, s_v \in \mathbb{R}$, opacity $o \in \mathbb{R}$, and color represented via spherical harmonics. A point in world space is parameterized in the local tangent frame as:

$$\mathbf{x}(u, v) = \boldsymbol{\mu} + s_u \mathbf{t}_u \, u + s_v \mathbf{t}_v \, v, \tag{1}$$

and the 2D Gaussian is evaluated as:

$$\mathcal{G}(u, v) = e^{-\frac{1}{2}(u^2+v^2)}, \tag{2}$$

where (u, v) are the local tangent-plane coordinates of the ray–surfel intersection. The pixel color $\mathbf{C}(\mathbf{p})$ is computed by alpha-compositing the ordered set of overlapping Gaussians:

$$\mathbf{C}(\mathbf{p}) = \sum_{i=1} \mathbf{c}_i \, \alpha_i \, \hat{\mathcal{G}}_i(u_i, v_i) \prod_{j=1}^{i-1} \Big(1 - \alpha_j \, \hat{\mathcal{G}}_j(u_j, v_j)\Big), \tag{3}$$

where $\mathbf{c}_i$ is the view-dependent color, α_i is the learned per-Gaussian opacity, and $\hat{\mathcal{G}}$ is a low-pass filtered kernel defined as $\hat{\mathcal{G}} = \max\{\mathcal{G}(u, v), \, \mathcal{G}_{2d}(\mathbf{p})\}$, where $\mathcal{G}_{2d}$ is a screen-space Gaussian filter that prevents aliasing at distant or oblique views. All attributes are optimized end-to-end through differentiable rasterization with photometric supervision from multi-view images.

3.2 Kernel function

The GS optimization processing could be seen as a signal fitting process. Unlike the transformation of frequency domain, GS does not have a complete basis functions such as the Fourier series and the wavelet transform. The primitive densify processing, split and clone, has to use smaller and narrower ellipsoids to compensate for the high frequency signal, which is the most important component of the spatial domain. However, the accumulation of those different-sized and scaled ellipsoids leads to the Gibbs phenomenon. The proposed deformable Gaussian kernel reshapes the kernel profile along each principal axis via a learnable warp function, enabling sharp boundary representation without Gibbs artifacts.

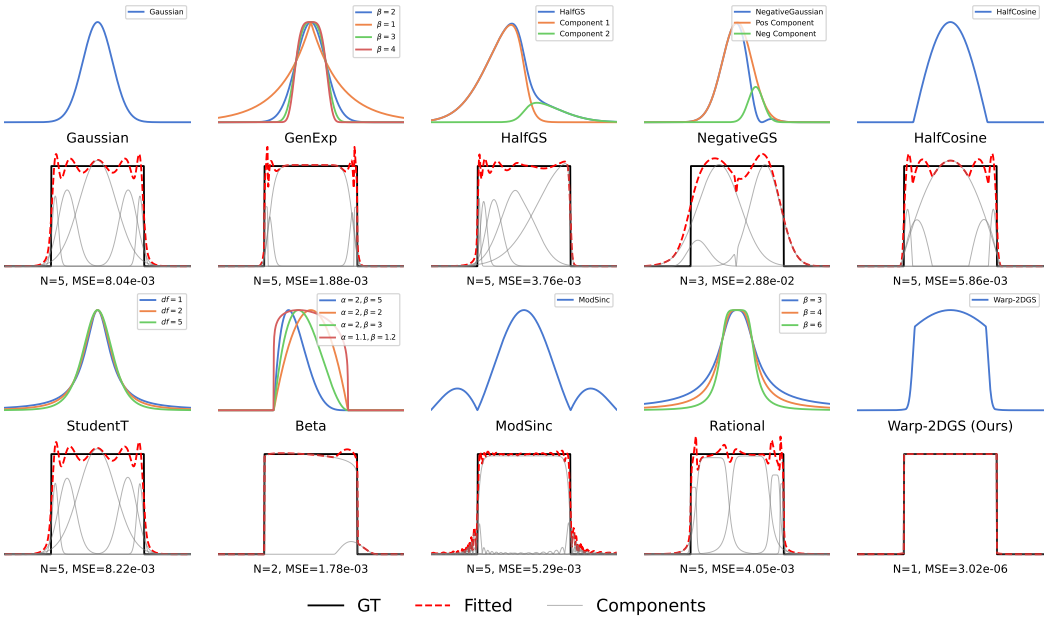


Figure 2: **Kernel function comparison and fitting results**. Odd rows show the primitives using different kernel functions, and even rows show the corresponding best-fitting results under 5 components. The red dashed line indicates the fitted curve. For our kernel, this 1D illustration uses a center-symmetric variant with two mirrored control points, whereas all reconstruction experiments use the single-knot kernel of Section 3.4.

As shown in Figure 2, we fit a 1D square wave defined as $f(x) = \mathbf{1}_{[0.25,\,0.75]}(x)$ using various kernel functions under a simplified 1D Gaussian reconstruction setting, which can be interpreted as observing the signal through a fixed-position orthographic camera from the top center, where contributions from up to 5 components are directly summed without alpha blending. For specific square wave fitting algorithms and all results, please refer to Appendix A. For each kernel type, we greedily search for the best combination rather than relying on clone-and-split. The results show that, except for the Beta function, all kernels tend to use the maximum number of components, and the Gibbs phenomenon persists across all tested kernels.

To prevent this phenomenon from happening, we choose a deformable mapping method to stretch the original Gaussian function to fit the square wave, where the sharp edges of it frequently appear in the scenes that need to be reconstructed.

3.3 Kernel Warping

The 1D deformed Gaussian is defined as $G(D(x)) = e^{-\frac{1}{2}D(x)^2}$. We extend this to 2D with axis-separated deformations $D_u(u)$ and $D_v(v)$ to avoid cross-axis distortion:

$$G(D(u,v)) = e^{-\frac{1}{2}(D_u(u)^2 + D_v(v)^2)}. \tag{4}$$

The design of this deformable mapping function should satisfy the following condition:

(1) The deformed function should remain smooth to maintain learnability like the original Gaussian, (2) the deformed function should not distort the Gaussian function too much, which would cause the function to introduce high-frequency fluctuations prematurely, and (3) the function should be able to recover the original Gaussian function during the training process. In other words, $D(t|\theta) \equiv t$ for some parameter setting.

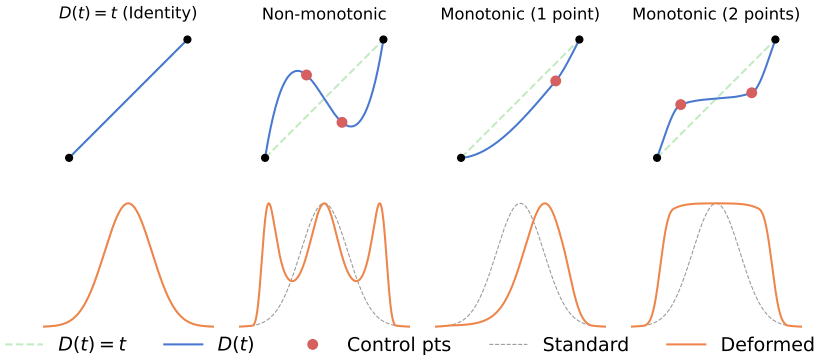


Figure 3: **Deformation functions and their corresponding deformed Gaussian kernels.** Top row: deformation function $D(t)$ with $t = x/r$. Bottom row: the resulting deformed Gaussian $G(r \cdot D(x/r))$, where r is a rescaling factor such that one unit in t corresponds to r standard deviations in the original domain.

Under these conditions, the simplest deformation function $D(t)$ should be monotonically increasing within the given bounds. As shown in Figure 3, we compare the effects of different deformation functions on the Gaussian kernel. The rescaling factor r normalizes the coordinate as $t = x/r$, so that one unit in t-space corresponds to r standard deviations of the Gaussian, allowing the deformation function to operate on a standardized scale range. A non-monotonic deformation produces multiple peaks in the deformed kernel, which not only can be equivalently represented by a mixture of standard Gaussians — undermining the shape–scale decoupling — but also introduces Gibbs-like ringing artifacts that our method is specifically designed to suppress. The last two columns demonstrate that adding more control points enables fitting higher-frequency signals, where the control points guide the model to capture sharp transitions while reducing overshooting near edges.

3.4 Deformation Function

The way to construct the deformation function is to use a piecewise linear (PWL) monotonic function. The endpoints are anchored at $(-1, -1)$ and $(1, 1)$ to preserve the identity mapping at the boundary. A single learnable interior knot $(c_1, d_1 = D(c_1))$ then fully specifies the piecewise-linear warp through three control points:

$$\{(c_i, d_i)\}_{i=0}^{2} = \big\{(-1,-1),\ (c_1, d_1),\ (1,1)\big\}. \tag{5}$$

with $-1 < c_1 < 1$ and $-1 < d_1 < 1$. The deformation function is

$$D(t|\mathbf{c}, \mathbf{d}) = \begin{cases} t, & |t| > 1, \\ d_i + \dfrac{d_{i+1} - d_i}{c_{i+1} - c_i}\,(t - c_i), & t \in [c_i,\ c_{i+1}], \quad i = 0, 1. \end{cases} \tag{6}$$

Forward pass. For $t \in [c_i, c_{i+1}]$, define the interval length h_i, the vertical displacement Δd_i, and the local coordinate o:

$$h_i = c_{i+1} - c_i, \qquad \Delta d_i = d_{i+1} - d_i, \qquad o = \frac{t - c_i}{h_i}.$$

Then the PWL interpolation can be written as

$$D(t) = (1 - o)d_i + o d_{i+1}. \tag{7}$$

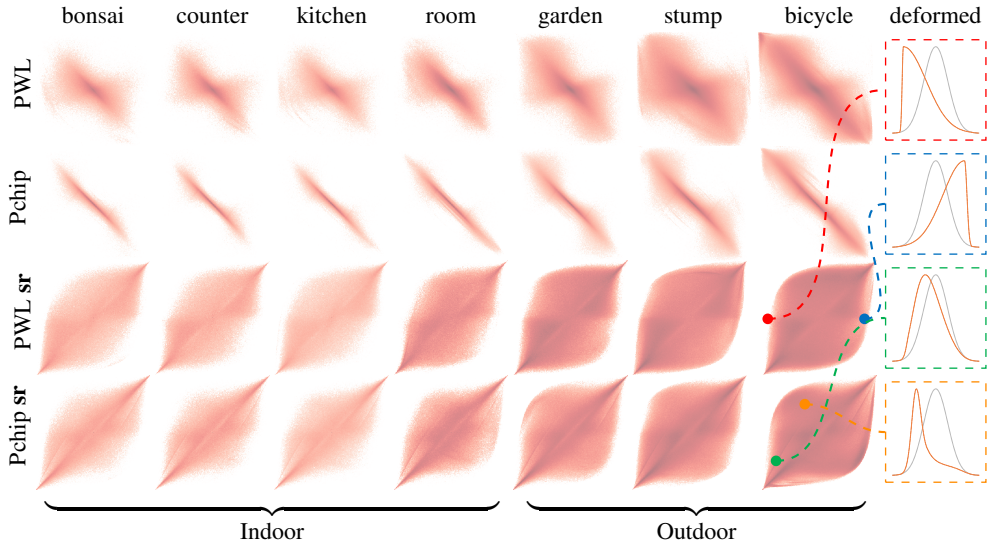


Figure 4: **Control point distributions across MipNeRF 360 scenes.** 2D histogram of (c, d) coordinates along the u direction across all scenes in the MipNeRF 360 dataset ($\times 8$ downsampling). The **sr** suffix denotes the slope-ratio parameterization. Colored markers highlight representative (c, d) positions; dashed boxes on the right show the corresponding deformed Gaussian kernels (orange) against the standard Gaussian (gray).

Local backward pass. The gradients with respect to t and the interior control point (c_1, d_1) are:

$$\frac{\partial D}{\partial t} = \frac{\Delta d_i}{h_i}, \quad \frac{\partial D}{\partial c_1} = \begin{cases} -o\,\frac{\Delta d_0}{h_0}, & t \in [c_0, c_1], \\ -(1-o)\,\frac{\Delta d_1}{h_1}, & t \in [c_1, c_2], \end{cases} \quad \frac{\partial D}{\partial d_1} = \begin{cases} o, & t \in [c_0, c_1], \\ 1-o, & t \in [c_1, c_2]. \end{cases} \tag{8}$$

These partial derivatives are independent of how the interior control point is parameterized. Different parameterizations only change the chain rule from the learnable parameters to (c_1, d_1).

3.4.1 Direct Parameterization

A direct parameterization learns the horizontal and vertical coordinates of the interior control point independently. Let $a, b \in \mathbb{R}$ be two learnable scalars. We map them to $(-1, 1)$ by

$$c_1 = 2\sigma(a) - 1, \qquad d_1 = 2\sigma(b) - 1, \tag{9}$$

where $\sigma(\cdot)$ denotes the sigmoid function. This automatically ensures $-1 < c_1 < 1$ and $-1 < d_1 < 1$ without sorting or clamping.

The chain rule gives

$$\frac{\partial G}{\partial a} = \frac{\partial G}{\partial D} \cdot \frac{\partial D}{\partial c_1} \cdot \frac{\partial c_1}{\partial a}, \qquad \frac{\partial G}{\partial b} = \frac{\partial G}{\partial D} \cdot \frac{\partial D}{\partial d_1} \cdot \frac{\partial d_1}{\partial b} \tag{10}$$

where $G(t) = r\,D(t/r)$, so that

$$\frac{\partial G}{\partial a} = r\,\frac{\partial D(t/r)}{\partial c_1}\,\frac{\partial c_1}{\partial a}, \qquad \frac{\partial G}{\partial b} = r\,\frac{\partial D(t/r)}{\partial d_1}\,\frac{\partial d_1}{\partial b}. \tag{11}$$

3.4.2 Slope-Ratio Parameterization

Although the direct control-point parameterization is straightforward, it can be unstable during optimization because the horizontal and vertical coordinates of the interior control points are strongly coupled through direct parameterization. In particular, for an increasing monotone deformation, the local gradients with respect to c_1 and d_1 tend to have opposite signs. This can be seen directly from the local gradient formulas above (8). The two gradients $\frac{\partial D}{\partial c_1}, \frac{\partial D}{\partial d_1}$ are linearly coupled by the positive local slope $\Delta d_i / h_i$ and therefore have opposite signs; near the origin, they are nearly negatives of

each other, causing the control point to move predominantly along the anti-diagonal direction during gradient descent. Figure 4 visualizes the distribution of learned control points in their parameter space. A broader spread indicates greater adaptability, while concentration along the anti-diagonal implies that the control points are strongly correlated, effectively restricting deformations to a low-dimensional subspace. Outdoor scenes consistently exhibit a wider spread than indoor ones, reflecting higher demand for complex kernel shapes. The sampled examples at the extremes further reveal asymmetric deformations associated with higher-frequency representations.

Moreover, the sensitivity is amplified when one of the PWL intervals becomes short. The segment length h_i appears in the denominator of the segment slope, and the gradient with respect to the knot location contains an even stronger dependence on the interval length, $\frac{\partial D}{\partial c_1} \propto \frac{\Delta d_i}{h_i^2}$. To decouple these effects, we instead parameterize the PWL deformation by the knot location c and the slope ratio $R = \frac{d_0}{h_0} / \frac{d_1}{h_1}$ between the two segments. Let $a, b \in \mathbb{R}$ be two learnable scalars and define

$$c_1 = 2\sigma(a) - 1, \qquad q = Q_{\max} \cdot (2\sigma(b) - 1), \qquad R = \exp(q) \tag{12}$$

where Q_{max} is set to control the maximum slope ratio, preventing excessively large slopes from causing numerical instability. An alternative would be to clamp the slope directly, but this introduces asymmetric gradients for a and b, effectively requiring different learning rates for the two parameters. To satisfy the endpoint constraints $D(-1) = -1$ and $D(1) = 1$, the slopes must satisfy

$$k_0 h_0 + k_1 h_1 = 2, \qquad k_0 = R k_1. \tag{13}$$

Solving these constraints gives the induced interior control point

$$d_1 = -1 + \frac{2Rh_0}{h_1 + Rh_0}. \tag{14}$$

The backward pass reuses the local PWL gradients derived above. We first compute gradients with respect to the induced control point (c_1, d_1), and then apply the chain rule.

3.5 Smooth Monotonic Deformation Function

Smooth, continuously differentiable functions are generally preferred in deep learning for their well-behaved gradient flow. We therefore implement a smooth variant using Pchip (Piecewise Cubic Hermite Interpolating Polynomial) [7] as the interpolator. Pchip is parameterized by the same set of control points as PWL, but replaces the linear segments with cubic polynomials that provide C^1 continuity and smoother gradient flow during backpropagation, while guaranteeing monotonicity by construction through the Fritsch–Carlson condition [7].

Pchip forward pass. Using the interval lengths h_i and slopes k_i, the derivatives at the two fixed endpoints are:

$$m_0 = \max\left(\frac{(2h_0 + h_1)\,k_0 - h_0\,k_1}{h_0 + h_1},\ 0\right), \qquad m_2 = \max\left(\frac{(2h_1 + h_0)\,k_1 - h_1\,k_0}{h_1 + h_0},\ 0\right). \tag{15}$$

The derivative at the interior point c_1 is the weighted harmonic mean of the adjacent slopes:

$$m_1 = \begin{cases} \dfrac{(w_1 + w_2)k_0 k_1}{w_1 k_1 + w_2 k_0}, & \text{if } k_0 k_1 > 0, \\ 0, & \text{otherwise,} \end{cases} \tag{16}$$

where $w_1 = 2h_1 + h_0$ and $w_2 = h_1 + 2h_0$ denote the corresponding weights. The output is:

$$D(t) = H_0(o)\,d_i + H_1(o)\,h_i\,m_i + H_2(o)\,d_{i+1} + H_3(o)\,h_i\,m_{i+1}, \quad t \in [x_i, x_{i+1}] \tag{17}$$

where the cubic Hermite basis functions are:

$$\begin{aligned} H_0(o) &= 2o^3 - 3o^2 + 1, & H_1(o) &= o^3 - 2o^2 + o, \\ H_2(o) &= -2o^3 + 3o^2, & H_3(o) &= o^3 - o^2. \end{aligned} \tag{18}$$

Outside the interpolation range, $D(t) = t$ for $|t| > 1$. The full derivation of the backward pass for Pchip is provided in Appendix G.

All these deformation functions modify only the 3D projected distance in $\mathcal{G}(u, v)$ for opacity calculation; depth is always computed from undeformed coordinates. Since the low-pass filter selects $\hat{\mathcal{G}} = \max\{\mathcal{G}(u, v), \mathcal{G}_{2d}(\mathbf{p})\}$ and $\mathcal{G}_{2d}$ is evaluated in screen space, the deformation is naturally bypassed whenever the filter activates.

3.6 Extent Computation

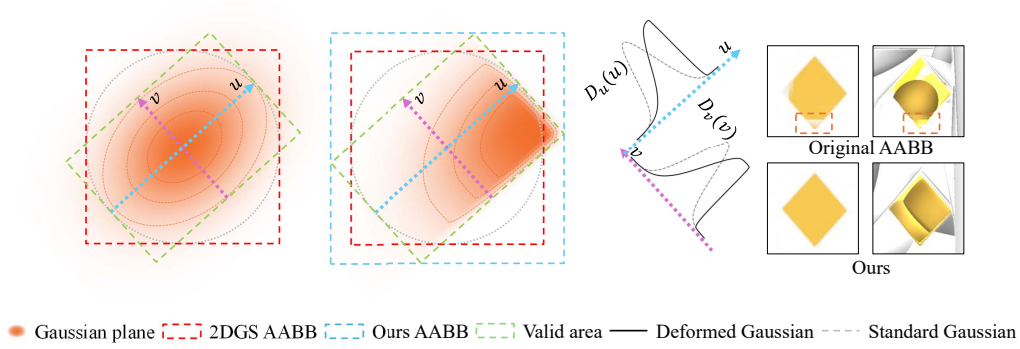


Figure 5: **AABB expansion for deformed kernels**. Left: The original 2DGS AABB (red) may clip the deformed Gaussian (orange); Middle: our expanded AABB (blue) fully covers the valid area (green). Right: deformation curves along the two principal axes, with zoomed-in comparisons of rendering with and without our AABB expansion.

In the rasterization stage [14], the original 2DGS computes the Axis-Aligned Bounding Box (AABB) from the maximum axis of the projected ellipse. This relies on the assumption that values beyond the 3σ boundary are negligible. However, this assumption no longer holds when the Gaussian kernel is deformed. As illustrated in Figure 5, in the extreme case where $D_u(u) = D_v(v) = 3$, the Mahalanobis distance at the AABB corner remains 3, leaving a non-negligible kernel value. Therefore, the AABB must be expanded to account for the deformation. Specifically, to prevent clipping artifacts, the cutoff value in this study should be multiplied by at least $\sqrt{2}$.

4 Experiments

4.1 Experimental Setup

All experiments are conducted on a single NVIDIA RTX 4090 GPU. We train for 30,000 iterations following the 2DGS training schedule with default hyperparameters. For our method, the deformation parameters use a rescale factor $r = 3$ and a learning rate of 0.0025, set to half of the scaling learning rate. Q_{max} is set to 3 to prevent computational overflow, $\exp(Q_{max}) \simeq 20$. We initialize $a = b = 0$ so that $\sigma(a) = \sigma(b) = 0.5$. This gives $c_1 = d_1 = 0$ and $R = 1$, and the deformation function reduces to the identity under both parameterizations. Same as 2DGS, we use the same optimizer, Adam [19], for other original parameters. To avoid affecting densification, we compute the radius using the original cutoff value 3 for the densification strategy, but use the extended cutoff $3 * \sqrt{2}$ for rasterization. For all baselines, we use their official implementations with default settings.

4.2 Main Comparison

We compare our method against several 2D-based Gaussian splatting methods: 2DGS [14], TriangleGS [13], and DRK-GS [15] (evaluated under three densification settings: dense, middle, and sparse, as defined in their original work). We also include 3D-based methods 3DHGS [21] and BetaGS [22] as reference. All methods are evaluated on Mip-NeRF 360 [1] ($\times 4$, $\times 8$) and Tanks & Temples [20] (COLMAP-processed by PixelGS [33]), reporting PSNR, SSIM, and LPIPS.

As shown in Table 1, our method consistently outperforms 2DGS across all dataset splits and metrics, with particularly notable gains on indoor scenes, and demonstrates more balanced performance across resolutions and scene types than TriangleGS. Against 3D methods (3DHGS, BetaGS), our approach narrows the gap, showing that our kernels can effectively enhance rendering quality within the 2D framework. Per-scene results and training time are provided in Appendix B, and qualitative comparisons in Appendix F.

Table 1: **Quantitative comparison on the Mip-NeRF 360 and Tanks & Temples datasets.** Outdoor and Indoor splits are averaged over their respective scenes. best and second among 2DGS methods. **sr**: slope parameterization method. †: 3DGS methods. * Results taken from the original paper. For DRK-GS, only dense is reported on TNT. 3DHGS omitted as full-scene results are unavailable.

	MipNeRF360 ×4						MipNeRF360 ×8						TNT ×1		
	Outdoor			Indoor			Outdoor			Indoor					
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
2DGS	25.85	0.778	0.259	30.51	0.929	0.169	27.03	0.835	0.166	31.19	0.945	0.093	23.66	0.832	0.280
TriangleGS	25.89	0.787	0.203	28.27	0.902	0.167	26.56	0.823	0.168	29.05	0.913	0.117	23.25	0.832	0.239
DRK-GS (sparse)	23.78	0.678	0.351	27.20	0.875	0.215	24.63	0.745	0.262	27.82	0.903	0.133	×	×	×
DRK-GS (middle)	24.12	0.703	0.320	27.47	0.882	0.207	24.94	0.762	0.238	27.78	0.903	0.131	×	×	×
DRK-GS (dense)	24.47	0.729	0.285	27.62	0.885	0.199	25.15	0.778	0.213	27.82	0.904	0.128	22.20	0.794	0.310
Ours (PWL)	25.93	0.784	0.243	30.85	0.933	0.160	27.08	0.838	0.157	31.36	0.947	0.088	23.86	0.835	0.272
Ours (Pchip)	25.89	0.782	0.246	30.87	0.933	0.161	27.06	0.837	0.159	31.30	0.947	0.089	23.91	0.835	0.274
Ours (PWL-**sr**)	25.94	0.784	0.245	30.89	0.933	0.160	27.11	0.839	0.157	31.43	0.948	0.088	23.92	0.836	0.273
Ours (Pchip-**sr**)	25.87	0.782	0.248	30.88	0.932	0.161	27.07	0.838	0.159	31.33	0.946	0.089	23.91	0.835	0.275
3DHGS†	26.40	0.798	0.175	32.54	0.948	0.093	27.83	0.857	0.111	33.29	0.962	0.046	–	–	–
BetaGS†	26.55	0.817	0.177	32.65	0.950	0.122	27.70	0.859	0.120	33.50	0.964	0.060	24.85*	0.870*	0.140*

4.3 PSNR vs. # of Gaussians

To explore the quality–efficiency trade-off of our method, we vary the number of Gaussians by terminating densification early and compare against 2DGS under the same settings. In theory, the Gibbs phenomenon can be alleviated by increasing the number of basis functions, which in our case corresponds to using more Gaussian primitives. Therefore, evaluating this trade-off is crucial for highlighting the effectiveness of the proposed method. As shown in Figure 6, our method consistently outperforms 2DGS across all Gaussian budgets. Notably, for indoor scenes, early termination of densification improves reconstruction quality, suggesting that the expressiveness of piecewise interpolation kernels effectively compensates for fewer primitives. Furthermore, the slope ratio (sr) parameterization provides additional gains in the low-Gaussian regime, consistent with its broader coverage in the control point space.

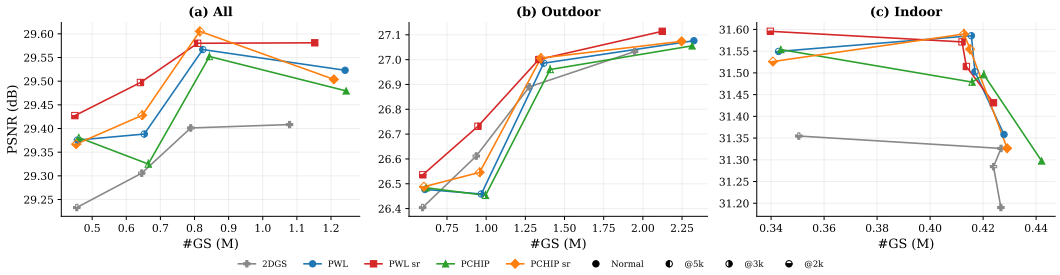


Figure 6: **PSNR vs. number of Gaussians on MipNeRF360 (×8)**. Markers denote densification schedules: Normal (15k iter), @5k, @3k, and @2k. Our methods achieve higher PSNR than 2DGS at comparable or fewer Gaussians. For indoor scenes (c), early densification termination improves quality, indicating that per-kernel expressiveness compensates for fewer primitives. Lines connect markers in schedule order; 2DGS indoor @3k exceeds @5k in count because pruning halts together with densification.

Our results also reveal that Pchip, despite offering a smoother deformation function, does not yield a notable improvement over PWL. This suggests that Gaussian reconstruction benefits more from the ability to represent asymmetric and high-frequency opacity variations than from smoothness. The slope-ratio parameterization improves PWL in PSNR across all splits, owing to its broader exploration of the control-point parameter space, which enables more diverse opacity shapes per primitive. For Pchip, however, it brings no consistent gain.

4.4 Geometry Evaluation

Sharper opacity kernels do not come at the cost of geometry. Table 2 shows that PWL-sr matches 2DGS in mean F1, and that all four variants improve on Truck while dropping on Ignatius. Truck is dominated by planar panels and sharp man-made edges, where an asymmetric falloff helps, whereas

Table 2: **F1 score on Tanks & Temples.** Meetingroom is excluded for all methods.

Method	Barn	Cat.	Court.	Ign.	Truck	Mean
2DGS	0.386	0.248	0.120	0.496	0.465	0.343
PWL	0.368	0.235	0.106	0.438	0.506	0.331
PWL-**sr**	0.380	0.249	0.131	0.469	0.508	0.347
Pchip	0.376	0.230	0.104	0.433	0.533	0.335
Pchip-**sr**	0.383	0.240	0.056	0.404	0.505	0.318

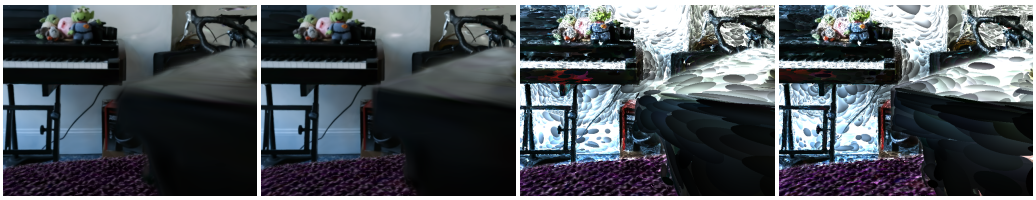
Table 3: **Boundary PSNR on Tanks & Temples.** Ours as differences to 2DGS.

Method	r=1	r=2	r=4	r=8	r=16	full
2DGS	18.53	19.32	20.20	21.12	21.96	23.32
PWL	+.130	+.151	+.173	+.190	+.211	+.239
PWL-**sr**	+.209	+.238	+.267	+.294	+.318	+.356
Pchip	+.133	+.161	+.187	+.208	+.225	+.278
Pchip-**sr**	+.034	+.035	+.035	+.045	+.067	+.107

the smooth curved surfaces of Ignatius are already well served by a symmetric Gaussian. We follow the 2DGS protocol and exclude Meetingroom for all methods because it exceeds our 64 GB host memory. The low Pchip-sr score on Courthouse comes from the ICP registration step of the official evaluation script, and we report it unchanged.

4.5 Boundary Evaluation

The gains of our kernel are not confined to smooth regions. Following the band protocol of Boundary IoU [4], we compute PSNR within bands around ground-truth Sobel edges dilated by radius r. Table 3 shows that every variant improves boundary PSNR over 2DGS at every radius, with PWL-sr remaining the strongest, and SSIM results are given in Appendix C. The gain in the narrowest bands is smaller than over the full image, since a boundary pixel is composited from many overlapping splats and alpha blending partly averages out the effect of sharpening any single kernel. Figure 7 visualizes the corresponding Gaussian ellipses. At object boundaries, 2DGS stacks many overlapping symmetric ellipses and blurs sharp edges, whereas our warped kernels align with boundary contours and produce cleaner edges.



2DGS RGB | PWL **sr** RGB (Ours) | 2DGS Ellipse | PWL **sr** Ellipse (Ours)

Figure 7: **Rendered images and Gaussian ellipse visualizations.** Our warped Gaussians align more tightly with object boundaries compared to the symmetric ellipses of 2DGS.

5 Discussion

Limitations. Our warp uses a single interior knot per axis. In the preliminary multi-knot experiments reported in Appendix D, adjacent knots tended to drift together and, once they coincided, failed to re-separate, a known degeneracy of free-knot splines [16, 2]. Progressive knot insertion is a promising remedy that we leave to future work. The axis-separated warp also cannot place a sharp transition along edges not aligned with the tangent axes, so residual ringing may persist at acute structures. Extending the kernel to 3DGS is tractable, since the PWL warp keeps the kernel Gaussian between consecutive knot planes, with the half-space integral of 3D-HGS [21] as the single-plane case. For dynamic scenes, existing 2DGS methods [29, 24] deform surfels with rotations only, and how warped kernels behave under non-rigid deformation remains unexplored. Finally, training takes roughly 35–45% longer than 2DGS, which tight bounding-box techniques such as Speedy-Splat [11] and FlashGS [6] could partly recover.

Conclusion. We present a learnable warped kernel for 2DGS that replaces the symmetric Gaussian opacity profile with axis-separated piecewise linear profiles. This design introduces asymmetric falloff while preserving differentiability, enabling sharp boundaries that symmetric smooth kernels cannot represent. Experiments on MipNeRF360 and Tanks & Temples show the best PSNR among 2DGS-based approaches with a comparable number of Gaussians.

Acknowledgments and Disclosure of Funding

This research was supported by the Government of Ontario through the Early Researcher Awards program (Award No. ER24-18-220). The authors declare no competing interests.

References

[1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5460–5469, New Orleans, LA, USA, June 2022. IEEE. ISBN 978-1-6654-6946-3. doi: 10.1109/CVPR52688.2022.00539. URL https://ieeexplore.ieee.org/document/9878829/.

[2] Gleb Beliakov. Least squares splines with free knots: global optimization approach. *Applied Mathematics and Computation*, 149(3):783–798, February 2004. ISSN 0096-3003. doi: 10.1016/S0096-3003(03)00179-6. URL https://www.sciencedirect.com/science/article/pii/S0096300303001796.

[3] Haodong Chen, Runnan Chen, Qiang Qu, Zhaoqing Wang, Tongliang Liu, Xiaoming Chen, and Yuk Ying Chung. Beyond Gaussians: Fast and High-Fidelity 3D Splatting with Linear Kernels, December 2024. URL http://arxiv.org/abs/2411.12440.

[4] Bowen Cheng, Ross Girshick, Piotr Doll{\'a}r, Alexander C. Berg, and Alexander Kirillov. Boundary IoU: Improving Object-Centric Image Segmentation Evaluation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15329–15337, Nashville, TN, USA, June 2021. IEEE. ISBN 978-1-6654-4509-2. doi: 10.1109/CVPR46437.2021.01508. URL https://ieeexplore.ieee.org/document/9578118/.

[5] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality Surface Reconstruction using Gaussian Surfels, April 2024. URL http://arxiv.org/abs/2404.17774.

[6] Guofeng Feng, Siyan Chen, Rong Fu, Zimu Liao, Yi Wang, Tao Liu, Boni Hu, Lining Xu, Zhilin Pei, Hengjie Li, Xiuhong Li, Ninghui Sun, Xingcheng Zhang, and Bo Dai. FlashGS: Efficient 3D Gaussian Splatting for Large-scale and High-resolution Rendering. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26652–26662, Nashville, TN, USA, June 2025. IEEE. ISBN 979-8-3315-4364-8. doi: 10.1109/CVPR52734.2025.02482. URL https://ieeexplore.ieee.org/document/11093809/.

[7] Frederick N Fritsch and Ralph E Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980.

[8] Kyle Gao, Liangzhi Li, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. Gaussian Building Mesh (GBM): Extract a Building's 3D Mesh with Google Earth and Gaussian Splatting. *Remote Sensing Applications: Society and Environment*, 40:101807, November 2025. ISSN 23529385. doi: 10.1016/j.rsase.2025.101807. URL http://arxiv.org/abs/2501.00625.

[9] David Gottlieb and Chi-Wang Shu. On the gibbs phenomenon and its resolution. *SIAM review*, 39(4):644–668, 1997.

[10] 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 *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19812–19822, Seattle, WA, USA, June 2024. IEEE. ISBN 979-8-3503-5300-6. doi: 10.1109/CVPR52733.2024.01873. URL https://ieeexplore.ieee.org/document/10656221/.

[11] Alex Hanson, Allen Tu, Geng Lin, Vasu Singla, Matthias Zwicker, and Tom Goldstein. Speedy-Splat: Fast 3D Gaussian Splatting with Sparse Pixels and Sparse Primitives. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21537–21546, Nashville, TN, USA, June 2025. IEEE. ISBN 979-8-3315-4364-8. doi: 10.1109/CVPR52734.2025.02006. URL https://ieeexplore.ieee.org/document/11094159/.

[12] Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Deliege, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3D Convex Splatting: Radiance Field Rendering with 3D Smooth Convexes. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21360–21369, Nashville, TN, USA, June 2025. IEEE. ISBN 979-8-3315-4364-8. doi: 10.1109/CVPR52734.2025.01990. URL https://ieeexplore.ieee.org/document/11094774/.

[13] Jan Held, Renaud Vandeghen, Adrien Deliege, Abdullah Hamdi, Silvio Giancola, Daniel Rebain, Anthony Cioppa, Bernard Ghanem, Andrea Vedaldi, Andrea Tagliasacchi, and Marc Van Droogenbroeck. Triangle Splatting for Real-Time Radiance Field Rendering. In *2026 International Conference on 3D Vision (3DV)*, pages 1248–1257, Vancouver, BC, Canada, March 2026. IEEE. ISBN 979-8-3315-7312-6. doi: 10.1109/3DV69130.2026.00123. URL https://ieeexplore.ieee.org/document/11533199/.

[14] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *Special Interest Group on GRAPHics and Interactive Techniques*. Association for Computing Machinery, June 2024. doi: 10.1145/3641519.3657428. URL http://arxiv.org/abs/2403.17888.

[15] Yi-Hua Huang, Ming-Xian Lin, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Deformable Radial Kernel Splatting. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21513–21523, Nashville, TN, USA, June 2025. IEEE. ISBN 979-8-3315-4364-8. doi: 10.1109/CVPR52734.2025.02004. URL https://ieeexplore.ieee.org/document/11092818/.

[16] David L. B. Jupp. Approximation to Data by Splines with Free Knots. *SIAM Journal on Numerical Analysis*, 15(2):328–343, April 1978. ISSN 0036-1429, 1095-7170. doi: 10.1137/0715022. URL http://epubs.siam.org/doi/10.1137/0715022.

[17] Artur Kasymov, Bartosz Czekaj, Marcin Mazur, Jacek Tabor, and Przemysław Spurek. NegGS: Negative Gaussian Splatting, May 2024. URL http://arxiv.org/abs/2405.18163.

[18] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4):1–14, July 2023. ISSN 0730-0301, 1557-7368. doi: 10.1145/3592433. URL https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017.

[21] Haolin Li, Jinyang Liu, Mario Sznaier, and Octavia Camps. 3D-HGS: 3D Half-Gaussian Splatting*. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10996–11005, Nashville, TN, USA, June 2025. IEEE. ISBN 979-8-3315-4364-8. doi: 10.1109/CVPR52734.2025.01027. URL https://ieeexplore.ieee.org/document/11093111/.

[22] Rong Liu, Dylan Sun, Meida Chen, Yue Wang, and Andrew Feng. Deformable Beta Splatting. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, SIGGRAPH Conference Papers '25, pages 1–11, New York, NY, USA, July 2025. Association for Computing Machinery. ISBN 979-8-4007-1540-2. doi: 10.1145/3721238.3730716. URL https://dl.acm.org/doi/10.1145/3721238.3730716.

[23] Rong Liu, Zhongpai Gao, Benjamin Planche, Meida Chen, Van Nguyen Nguyen, Meng Zheng, Anwesa Choudhuri, Terrence Chen, Yue Wang, Andrew Feng, and Ziyan Wu. Universal Beta Splatting, February 2026. URL http://arxiv.org/abs/2510.03312.

[24] Hidenobu Matsuki, Gwangbin Bae, and Andrew J. Davison. 4DTAM: Non-Rigid Tracking and Mapping via Dynamic Surface Gaussians. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26921–26932, Nashville, TN, USA, June 2025. IEEE. ISBN 979-8-3315-4364-8. doi: 10.1109/CVPR52734.2025.02507. URL https://ieeexplore.ieee.org/document/11093030/.

[25] Gyuhyeon Pak, Hae Min Cho, and Euntai Kim. G2S-ICP SLAM: Geometry-aware Gaussian Splatting ICP SLAM, July 2025. URL http://arxiv.org/abs/2507.18344.

[26] Hashiru Pramuditha, Vinasirajan Viruthshaan, Vishagar Arunan, Saeedha Nazar, Sameera Ramasinghe, Simon Lucey, and Ranga Rodrigo. DARB-Splatting: Generalizing Splatting with Decaying Anisotropic Radial Basis Functions. In *2026 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4504–4514, Tucson, AZ, USA, March 2026. IEEE. ISBN 979-8-3315-5511-5. doi: 10.1109/WACV61042.2026.00438. URL https://ieeexplore.ieee.org/document/11492092/.

[27] Sam Roweis. Levenberg-marquardt optimization. *Notes, University Of Toronto*, 52(1027):6, 1996.

[28] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on computer vision and pattern recognition (CVPR)*, 2016.

[29] Shuo Wang, Binbin Huang, Ruoyu Wang, and Shenghua Gao. Space-time 2D Gaussian Splatting for Accurate Surface Reconstruction under Complex Dynamic Scenes, September 2024. URL http://arxiv.org/abs/2409.18852.

[30] Minye Wu, Haizhao Dai, Kaixin Yao, Tinne Tuytelaars, and Jingyi Yu. BG-Triangle: Bézier Gaussian Triangle for 3D Vectorization and Rendering. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16197–16207, Nashville, TN, USA, June 2025. IEEE. ISBN 979-8-3315-4364-8. doi: 10.1109/CVPR52734.2025.01510. URL https://ieeexplore.ieee.org/document/11093081/.

[31] Qipeng Yan, Mingyang Sun, and Lihua Zhang. 2DGS-Avatar: Animatable High-fidelity Clothed Avatar via 2D Gaussian Splatting, March 2025. URL http://arxiv.org/abs/2503.02452.

[32] Yixin Yang, Yang Zhou, and Hui Huang. Introducing Unbiased Depth into 2D Gaussian Splatting for High-accuracy Surface Reconstruction. *Computer Graphics Forum*, 44(7):e70252, October 2025. ISSN 0167-7055, 1467-8659. doi: 10.1111/cgf.70252. URL https://onlinelibrary.wiley.com/doi/10.1111/cgf.70252.

[33] Zheng Zhang, Wenbo Hu, Yixing Lao, Tong He, and Hengshuang Zhao. Pixel-GS: Density Control with Pixel-aware Gradient for 3D Gaussian Splatting. In *Computer Vision – ECCV 2024*, pages 326–342, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-72655-2. doi: 10.1007/978-3-031-72655-2_19. URL http://arxiv.org/abs/2403.15530.

[34] Ziyu Zhang, Binbin Huang, Hanqing Jiang, Liyang Zhou, Xiaojun Xiang, and Shuhan Shen. Quadratic Gaussian Splatting: High Quality Surface Reconstruction with Second-Order Geometric Primitives. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 28260–28270, Honolulu, HI, USA, October 2025. IEEE. ISBN 979-8-3315-8775-8. doi: 10.1109/ICCV51701.2025.02624. URL https://ieeexplore.ieee.org/document/11445530/.

[35] Xingguang Zhong, Yue Pan, Liren Jin, Marija Popović, Jens Behley, and Cyrill Stachniss. Globally Consistent RGB-D SLAM With 2-D Gaussian Splatting. *IEEE Transactions on Robotics*, 42:2360–2380, 2026. ISSN 1552-3098, 1941-0468. doi: 10.1109/TRO.2026.3697159. URL https://ieeexplore.ieee.org/document/11535319/.

[36] Jialin Zhu, Jiangbei Yue, Feixiang He, and He Wang. 3D Student Splatting and Scooping. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21045–21054, Nashville, TN, USA, June 2025. IEEE. ISBN 979-8-3315-4364-8. doi: 10.1109/CVPR52734.2025.01960. URL https://ieeexplore.ieee.org/document/11092303/.

A Square wave fitting

To systematically evaluate each kernel's ability to represent sharp features, we design a greedy incremental fitting procedure that iteratively adds one kernel component at a time, initializing each at the dominant residual peak and refitting all parameters jointly via Levenberg–Marquardt optimization [27]. The Bayesian Information Criterion (BIC) governs model selection, balancing reconstruction accuracy against model complexity. The full procedure is summarized in Algorithm 1.

Algorithm 1 Greedy Incremental Gaussian Mixture Fitting

Require: Spectrum (x, y), kernel class $\mathcal{K}$, max components $N_{\max}$, BIC threshold δ
Ensure: Optimal number of components n^*, fitted parameters $\boldsymbol{\theta}^*$

$r \leftarrow y$ ▷ Initialize residual
$\mathrm{BIC}^* \leftarrow +\infty, \quad n^* \leftarrow 0, \quad \boldsymbol{\theta}^* \leftarrow \varnothing$
$\mathcal{C} \leftarrow \varnothing$ ▷ Set of kernel components
for $n = 1, 2, \ldots, N_{\max}$ **do**
  Detect peaks $\mathcal{P} \leftarrow \textsc{FindPeaks}(r)$ with height $\geq 0.1 \cdot \max(r)$
  if $\mathcal{P} = \varnothing$ **then**
    $\mathcal{P} \leftarrow \{\arg\max_i \, r_i\}$
  end if
  $j^* \leftarrow \arg\max_{j \in \mathcal{P}} \, r_j$ ▷ Select dominant residual peak
  Initialize new kernel: $\mu \leftarrow x_{j^*}, \quad a \leftarrow r_{j^*}, \quad \sigma \leftarrow \frac{x_{\max} - x_{\min}}{4n}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathcal{K}(\mu, \sigma, a)\}$
  $\boldsymbol{\theta}_0 \leftarrow$ concatenate parameters from all kernels in $\mathcal{C}$
  $\boldsymbol{\theta} \leftarrow \textsc{LevenbergMarquardt}\Big(\min_{\boldsymbol{\theta}} \left\| y - \sum_{k \in \mathcal{C}} k(x; \boldsymbol{\theta}) \right\|^2, \; \boldsymbol{\theta}_0\Big)$
  $\hat{y} \leftarrow \sum_{k \in \mathcal{C}} k(x; \boldsymbol{\theta})$
  $\mathrm{RSS} \leftarrow \sum_i (y_i - \hat{y}_i)^2$
  $\mathrm{BIC} \leftarrow N \ln\left(\frac{\mathrm{RSS}}{N}\right) + |\boldsymbol{\theta}| \ln N$ ▷ Bayesian Information Criterion
  $r \leftarrow y - \hat{y}$ ▷ Update residual
  if $\mathrm{BIC} < \mathrm{BIC}^* - \delta$ **then** ▷ Accept only if BIC improves
    $\mathrm{BIC}^* \leftarrow \mathrm{BIC}, \quad \boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}, \quad n^* \leftarrow n$
  end if
end for
return $n^*, \boldsymbol{\theta}^*$

Figure 8 presents the results for each kernel under identical conditions. As the number of components increases, symmetric kernels consistently exhibit Gibbs-like oscillations near the discontinuities at $x = 0.25$ and $x = 0.75$. In contrast, the proposed our warped Gaussian kernel suppresses these ringing artifacts even with few components, as its asymmetric profiles can conform to sharp transitions without compensatory oscillation from neighboring primitives.

B Per-Scene Quantitative Results

Table 4 reports per-scene results on MipNeRF 360 at $\times 4$ and $\times 8$ resolutions. Our variants achieve the best or second-best scores among 2D methods across nearly all scenes and metrics. The training time of our method lies between 2DGS and the other kernel-based methods, and remains substantially lower than that of DRK-GS and TriangleGS.

C Boundary Evaluation

Table 5 complements the boundary PSNR in Table 3 with SSIM computed over the same bands. PWL and PWL-sr improve boundary SSIM at every radius, whereas Pchip and Pchip-sr fall slightly below 2DGS.

D Multi-Knot Variants

In early exploration, we tested a range of multi-knot parameterizations for both PWL and Pchip, shown in Figure 9. Monotonicity requires the interior knots to stay strictly ordered, so each variant needs its own way

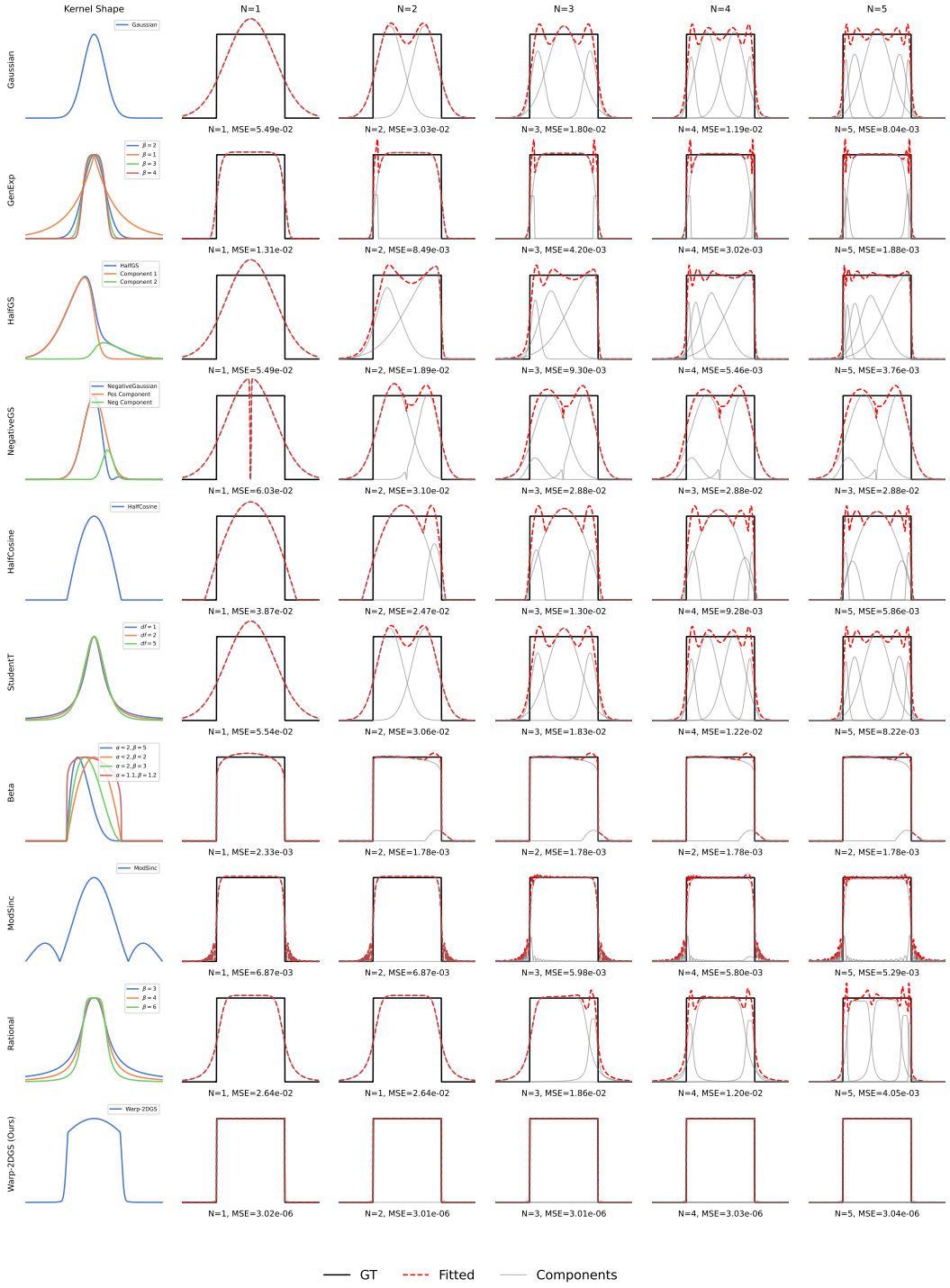


Figure 8: **Qualitative comparison of kernel functions for 1D square wave signal fitting.** Each row corresponds to a different kernel type. The first column illustrates the kernel shape under representative parameterizations. The remaining columns show the fitting results on a square wave target signal using an increasing number of components ($N = 1, 2, 3, 4, 5$), where the black curve denotes the ground truth, the red dashed curve denotes the overall fit, and gray curves denote individual components. Notably, our warped Gaussian kernel exhibits nearly no Gibbs-like ringing artifacts near discontinuities, even with a small number of components. Our kernel uses the same center-symmetric variant as in Figure 2.

Table 4: **Per-scene results on Mip-NeRF 360.** Best and second among 2D methods. **sr**: slope parameterization method. †: 3D methods. Train: average training time (min).

Method	PSNR↑							SSIM↑							LPIPS↓							Train
	Bicy.	Bons.	Coun.	Gard.	Kitc.	Room	Stump	Bicy.	Bons.	Coun.	Gard.	Kitc.	Room	Stump	Bicy.	Bons.	Coun.	Gard.	Kitc.	Room	Stump	
×4 resolution																						
2DGS	24.75	31.65	28.55	26.68	30.38	31.48	26.11	0.734	0.945	0.906	0.843	0.930	0.933	0.756	0.305	0.170	0.185	0.168	0.114	0.206	0.303	13.6
TriangleGS	24.56	28.15	27.58	27.10	28.75	28.59	26.01	0.752	0.897	0.895	0.860	0.915	0.903	0.750	0.227	0.187	0.165	0.123	0.128	0.186	0.257	29.9
DRK-GS (sparse)	23.15	27.87	26.60	23.97	26.03	28.30	24.23	0.631	0.888	0.864	0.744	0.855	0.894	0.660	0.388	0.217	0.223	0.263	0.172	0.250	0.401	68.7
DRK-GS (middle)	23.43	28.36	26.76	24.41	26.43	28.33	24.51	0.660	0.902	0.868	0.766	0.864	0.895	0.684	0.355	0.201	0.217	0.237	0.164	0.244	0.370	74.9
DRK-GS (dense)	23.77	28.15	26.96	24.99	26.75	28.65	24.63	0.691	0.895	0.873	0.790	0.869	0.902	0.707	0.316	0.197	0.208	0.205	0.159	0.233	0.334	90.4
Ours (PWL)	24.74	32.04	28.73	26.80	31.23	31.41	26.24	0.741	0.949	0.910	0.848	0.939	0.934	0.763	0.286	0.159	0.177	0.155	0.103	0.199	0.288	19.6
Ours (Pchip)	24.77	32.04	28.75	26.77	31.19	31.51	26.13	0.741	0.948	0.909	0.846	0.939	0.934	0.760	0.289	0.162	0.179	0.157	0.103	0.201	0.292	21.9
Ours (PWL-**sr**)	24.77	32.08	28.74	26.84	31.18	31.57	26.20	0.742	0.949	0.910	0.848	0.939	0.935	0.763	0.289	0.160	0.177	0.156	0.103	0.199	0.290	19.9
Ours (Pchip-**sr**)	24.76	32.11	28.74	26.71	31.17	31.51	26.15	0.740	0.949	0.909	0.846	0.938	0.934	0.761	0.292	0.161	0.179	0.159	0.104	0.202	0.292	21.8
3DHGS†	25.28	33.98	30.15	27.56	33.09	32.93	26.35	0.767	0.962	0.928	0.867	0.952	0.949	0.759	0.200	0.093	0.106	0.105	0.063	0.108	0.219	19.3
BetaGS†	25.28	34.18	30.50	27.58	33.05	32.89	26.80	0.784	0.966	0.935	0.870	0.952	0.949	0.796	0.198	0.118	0.131	0.116	0.085	0.154	0.216	23.2
×8 resolution																						
2DGS	26.20	32.02	29.03	28.25	31.60	32.11	26.65	0.814	0.956	0.921	0.897	0.955	0.950	0.795	0.189	0.088	0.111	0.095	0.059	0.113	0.215	6.6
TriangleGS	25.90	28.66	28.28	27.30	30.03	29.23	26.48	0.798	0.900	0.906	0.884	0.933	0.913	0.788	0.189	0.133	0.117	0.103	0.085	0.134	0.212	15.1
DRK-GS (sparse)	24.38	28.67	26.97	25.12	26.98	28.65	24.41	0.723	0.917	0.886	0.816	0.896	0.914	0.695	0.282	0.122	0.145	0.172	0.108	0.156	0.332	35.2
DRK-GS (middle)	24.61	28.65	27.06	25.43	26.73	28.68	24.77	0.743	0.917	0.888	0.828	0.889	0.915	0.716	0.255	0.120	0.142	0.157	0.111	0.152	0.302	37.5
DRK-GS (dense)	24.84	28.31	27.15	25.89	27.07	28.76	24.73	0.762	0.908	0.891	0.844	0.897	0.919	0.729	0.224	0.121	0.138	0.138	0.105	0.147	0.278	45.2
Ours (PWL)	26.18	32.41	29.27	28.34	31.88	31.87	26.71	0.817	0.958	0.924	0.899	0.956	0.949	0.798	0.177	0.082	0.104	0.090	0.056	0.112	0.205	8.9
Ours (Pchip)	26.14	32.38	29.23	28.35	31.84	31.74	26.67	0.816	0.958	0.924	0.898	0.956	0.948	0.797	0.178	0.083	0.104	0.092	0.056	0.113	0.206	9.9
Ours (PWL-**sr**)	26.21	32.38	29.27	28.43	31.90	32.17	26.70	0.818	0.958	0.925	0.900	0.956	0.952	0.798	0.177	0.082	0.104	0.090	0.056	0.108	0.205	8.9
Ours (Pchip-**sr**)	26.19	32.41	29.28	28.33	31.67	31.94	26.70	0.817	0.958	0.925	0.899	0.956	0.948	0.798	0.180	0.083	0.104	0.092	0.057	0.114	0.206	10.2
3DHGS†	26.88	34.45	30.73	29.61	34.32	33.67	27.00	0.846	0.972	0.942	0.922	0.971	0.965	0.804	0.120	0.041	0.062	0.057	0.030	0.051	0.156	11.9
BetaGS†	26.70	34.65	30.96	29.27	34.42	33.96	27.14	0.841	0.973	0.945	0.914	0.971	0.968	0.821	0.126	0.052	0.072	0.068	0.039	0.076	0.167	19.6

Table 5: **Boundary SSIM on Tanks & Temples.** The 2DGS row is absolute, and the other rows are differences to 2DGS.

Method	r=1	r=2	r=4	r=8	r=16	full
2DGS	0.7538	0.7462	0.7447	0.7591	0.7785	0.8116
PWL	+0.0029	+0.0029	+0.0027	+0.0022	+0.0020	+0.0022
PWL-**sr**	+0.0067	+0.0064	+0.0057	+0.0050	+0.0045	+0.0043
Pchip	−0.0005	−0.0006	−0.0010	−0.0015	−0.0017	−0.0011
Pchip-**sr**	−0.0037	−0.0034	−0.0030	−0.0027	−0.0021	−0.0009

of enforcing this ordering. PWL2 adds a second learnable knot, and its variants differ only in how the ordering is enforced. PWL3 places a fixed knot at the origin with one learnable knot on each side.

These variants confound knot count with parameterization, and several were run under an earlier version of our pipeline, so we report qualitative trends rather than a controlled ablation. Subject to these caveats, none of the variants offered a clear advantage over the single-knot design. In PWL3, the two learnable knots lie on either side of a fixed origin and cannot collide with each other. PWL3-symmetric mirrors a single (c_1, d_1) pair, so the added knots neither split the gradient signal across more parameters nor allow two knots to collide. It therefore isolates the effect of the extra segments alone, and it performs on par with the single-knot design. PWL3-asymmetric and PWL3-asymmetric-y0 have more free parameters and improve over the 2DGS baseline, but fall short of the single-knot design. In the PWL2 family, two free interior knots lie in the same interval, and we observed adjacent knots drifting together and, once they coincided, failing to re-separate, with unstable gradients as they approached each other. The last two rows of Table 6 report PWL2-mult and PWL2-segment under the current pipeline.

This failure mode is well documented in the free-knot spline literature. Merged knots form degenerate faces of the feasible region on which the gradient separating them vanishes identically [16], and the number of such faces grows with the knot count. Local optimizers are reported to converge to these degenerate solutions rather than to the true optimum, particularly from uniformly spaced initializations [2]. A viable remedy is progressive knot insertion, which trains to convergence with one knot and then inserts additional knots along the learned curve. For PWL kernels this insertion is an exact reparameterization that leaves the curve unchanged. Adopting it introduces further design choices, such as when to insert new knots and where to place them, and we leave a systematic investigation to future work.

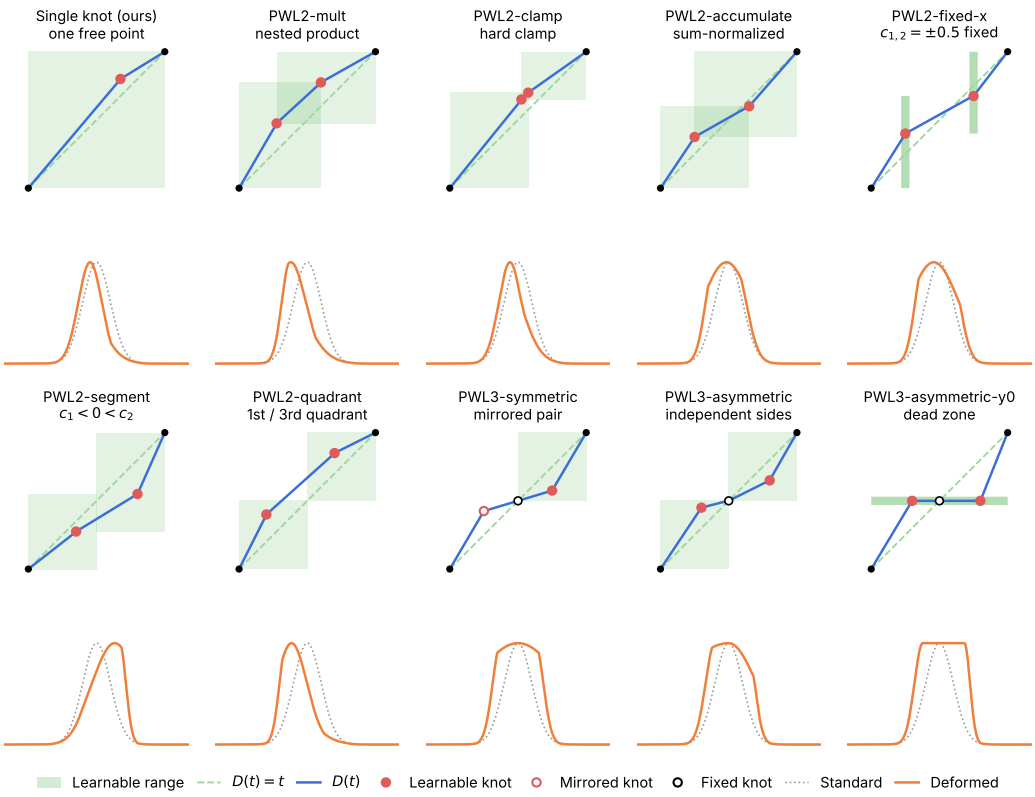


Figure 9: **Multi-knot deformation variants.** For each variant, the top row shows the deformation function $D(t)$ and the bottom row shows the resulting kernel $G(r \cdot D(x/r))$. Green shading marks the range in which each learnable knot can move while the other knots are held fixed. Parameter values are chosen for illustration and are not learned values.

E Rescale Factor

The rescale factor r sets the domain $G(-r, r)$ over which the control points can reshape the Gaussian. At $r = 1$ only the region near the peak is deformable, and a larger r extends the deformable region into the tail. Since the deformation function satisfies $D(t) = t$ for $|t| > 1$ (Eq. 6), r controls the size of the deformable region relative to the Gaussian, whose own extent is set independently by the scaling parameters. Too large an r risks introducing invalid area, while too small an r leaves insufficient room for deformation. As $r \to 0$, nearly all t/r exceed 1 and the kernel degenerates back to a standard Gaussian. We test $r \in \{2, 3, 4\}$ for both the direct and slope-ratio parameterizations of PWL on Mip-NeRF 360 ($\times 8$). As shown in Table 6, neither $r = 2$ nor $r = 4$ significantly outperforms our default $r = 3$, in either outdoor or indoor scenes.

F Qualitative Comparison

Figure 10 shows qualitative comparisons on selected scenes from MipNeRF 360. TriangleGS and DRK-GS tend to produce blurry or noisy textures in fine-detail regions, while 2DGS preserves overall structure but over-smooths sharp edges. Our method recovers sharper boundaries and finer textures, consistent with the quantitative improvements reported in Table 4.

G Pchip Backward Pass Derivation

We derive the gradients of the Pchip deformation function with respect to the learnable parameters a and b, where $c_1 = 2\sigma(a) - 1$ and $d_1 = 2\sigma(b) - 1$. Given the upstream gradient $\frac{\partial \mathcal{L}}{\partial D}$, the chain rule gives

$$\frac{\partial \mathcal{L}}{\partial a} = \frac{\partial \mathcal{L}}{\partial D} \frac{\partial D}{\partial c_1} \frac{\partial c_1}{\partial a}, \qquad \frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial D} \frac{\partial D}{\partial d_1} \frac{\partial d_1}{\partial b}. \tag{19}$$

Table 6: **Sensitivity to the rescale factor and the number of knots on Mip-NeRF 360 ($\times 8$).** $Q_{max} = 3$ for all rows. The one-knot rows with $r = 3$ are the configurations used in the main paper.

				Outdoor			Indoor		
Kernel	Knots	r	Param.	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
PWL	1	3	Direct	27.08	0.838	0.157	31.36	0.947	0.088
PWL	1	2	Direct	27.08	0.839	0.155	31.23	0.946	0.088
PWL	1	4	Direct	27.03	0.836	0.161	31.38	0.947	0.089
PWL	1	3	**sr**	27.11	0.839	0.157	31.43	0.948	0.088
PWL	1	2	**sr**	27.08	0.839	0.156	31.30	0.947	0.089
PWL	1	4	**sr**	27.03	0.836	0.161	31.39	0.947	0.089
PWL	2	3	mult	26.99	0.836	0.156	30.94	0.943	0.090
PWL	2	3	segment	26.96	0.836	0.158	31.14	0.945	0.090

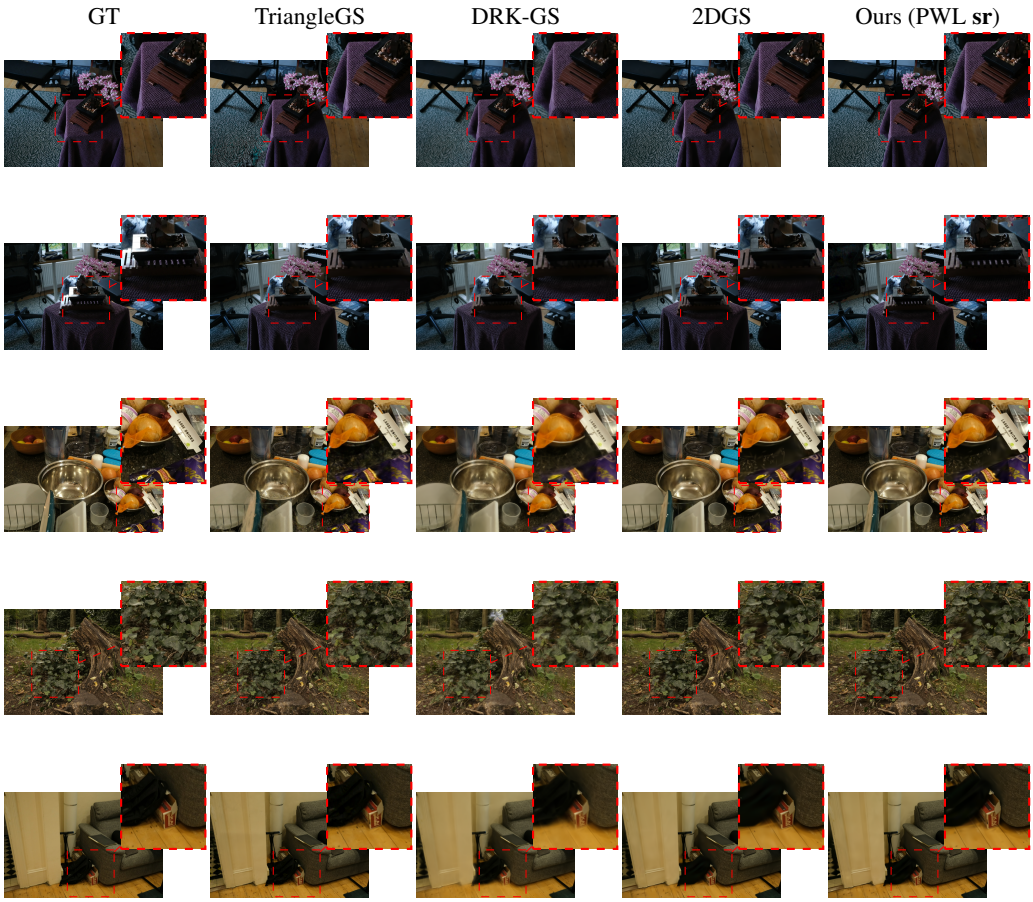


Figure 10: **Qualitative comparison on Mip-NeRF 360 ($\times 4$).** Red dashed boxes indicate the zoomed regions.

Below we derive $\frac{\partial D}{\partial t}$, $\frac{\partial D}{\partial c_1}$, and $\frac{\partial D}{\partial d_1}$. The final sigmoid chain-rule factors are $\frac{\partial c_1}{\partial a} = 2\sigma(a)(1-\sigma(a))$ and $\frac{\partial d_1}{\partial b} = 2\sigma(b)(1-\sigma(b))$.

Notation. The three control points are $\{(-1,-1),\ (c_1,d_1),\ (1,1)\}$. The interval lengths, slopes, and local coordinates on segment$[c_i, c_{i+1}]$ are

$$h_0 = c_1 + 1, \quad h_1 = 1 - c_1, \quad h_0 + h_1 = 2,$$
$$k_0 = \frac{d_1 + 1}{h_0}, \quad k_1 = \frac{1 - d_1}{h_1}, \qquad o = \frac{t - c_i}{h_i}.$$

The cubic Hermite interpolation is

$$D(t) = H_0(o)\, d_i + H_1(o)\, h_i\, m_i + H_2(o)\, d_{i+1} + H_3(o)\, h_i\, m_{i+1}, \quad t \in [c_i, c_{i+1}] \tag{20}$$

with basis functions $H_0 = 2o^3 - 3o^2 + 1$, $H_1 = o^3 - 2o^2 + o$, $H_2 = -2o^3 + 3o^2$, $H_3 = o^3 - o^2$, and their derivatives

$$\begin{aligned} H_0' &= 6o^2 - 6o, & H_1' &= 3o^2 - 4o + 1, \\ H_2' &= -6o^2 + 6o, & H_3' &= 3o^2 - 2o. \end{aligned} \tag{21}$$

Weights for the interior derivative: $w_1 = 2h_1 + h_0$, $w_2 = h_1 + 2h_0$, so $w_1 + w_2 = 3(h_0 + h_1) = 6$.

Step 1: Gradient with respect to t. Since $\partial o/\partial t = 1/h_i$:

$$\frac{\partial D}{\partial t} = \frac{1}{h_i}\frac{\partial D}{\partial o}, \qquad \frac{\partial D}{\partial o} = H_0'\, d_i + H_1'\, h_i\, m_i + H_2'\, d_{i+1} + H_3'\, h_i\, m_{i+1}. \tag{22}$$

Step 2: Gradients of k_0 and k_1. From $k_0 = (d_1 + 1)/h_0$ and $k_1 = (1 - d_1)/h_1$, using $\partial h_0/\partial c_1 = 1$ and $\partial h_1/\partial c_1 = -1$:

$$\begin{aligned} \frac{\partial k_0}{\partial c_1} &= -\frac{k_0}{h_0}, & \frac{\partial k_0}{\partial d_1} &= \frac{1}{h_0}, \\ \frac{\partial k_1}{\partial c_1} &= \frac{k_1}{h_1}, & \frac{\partial k_1}{\partial d_1} &= -\frac{1}{h_1}. \end{aligned} \tag{23}$$

Step 3: Gradients of m_0 and m_2 (endpoint derivatives). The unclamped values are

$$\tilde{m}_0 = \frac{(2h_0 + h_1)\, k_0 - h_0\, k_1}{2}, \qquad \tilde{m}_2 = \frac{(2h_1 + h_0)\, k_1 - h_1\, k_0}{2}.$$

Case $\tilde{m}_0 \geq 0$ (clamping inactive, $m_0 = \tilde{m}_0$). Let $N_0 = (2h_0 + h_1)k_0 - h_0 k_1$.

Gradient w.r.t. c_1:

$$\begin{aligned} \frac{\partial N_0}{\partial c_1} &= \underbrace{\frac{\partial(2h_0 + h_1)}{\partial c_1}}_{=1} k_0 + (2h_0 + h_1)\left(-\frac{k_0}{h_0}\right) - \underbrace{\frac{\partial h_0}{\partial c_1}}_{=1} k_1 - h_0\, \frac{k_1}{h_1} \\ &= k_0 - \frac{(2h_0 + h_1)}{h_0}\, k_0 - k_1 - \frac{h_0}{h_1}\, k_1 \\ &= k_0\left(1 - 2 - \frac{h_1}{h_0}\right) - k_1\left(1 + \frac{h_0}{h_1}\right) \\ &= -\frac{h_0 + h_1}{h_0}\, k_0 - \frac{h_0 + h_1}{h_1}\, k_1 = -2\left(\frac{k_0}{h_0} + \frac{k_1}{h_1}\right). \end{aligned} \tag{24}$$

Therefore

$$\frac{\partial m_0}{\partial c_1} = \frac{1}{2}\frac{\partial N_0}{\partial c_1} = -\frac{k_0}{h_0} - \frac{k_1}{h_1}. \tag{25}$$

Gradient w.r.t. d_1:

$$\frac{\partial N_0}{\partial d_1} = (2h_0 + h_1)\, \frac{1}{h_0} + h_0\, \frac{1}{h_1} = 2 + \frac{h_1}{h_0} + \frac{h_0}{h_1}. \tag{26}$$

Therefore

$$\frac{\partial m_0}{\partial d_1} = 1 + \frac{h_0^2 + h_1^2}{2\, h_0\, h_1}. \tag{27}$$

Case $\tilde{m}_0 < 0$ (clamping active, $m_0 = 0$): $\frac{\partial m_0}{\partial c_1} = \frac{\partial m_0}{\partial d_1} = 0$.

By the symmetry of $\tilde{m}_2$, when clamping is inactive ($\tilde{m}_2 \geq 0$):

$$\frac{\partial m_2}{\partial c_1} = \frac{k_0}{h_0} + \frac{k_1}{h_1} = -\frac{\partial m_0}{\partial c_1}, \qquad \frac{\partial m_2}{\partial d_1} = -1 - \frac{h_0^2 + h_1^2}{2\, h_0\, h_1} = -\frac{\partial m_0}{\partial d_1}. \tag{28}$$

Step 4: Gradients of m_1 (interior derivative). **Case $k_0k_1 \le 0$:** $m_1 = 0$, all gradients vanish.

Case $k_0k_1 > 0$: Define $\tau = w_1/k_0 + w_2/k_1$, so $m_1 = (w_1 + w_2)/\tau = 6/\tau$. Since $w_1 + w_2 = 6$ is constant:

$$\frac{\partial m_1}{\partial \theta} = -\frac{6}{\tau^2}\,\frac{\partial \tau}{\partial \theta} = -\frac{m_1^2}{6}\,\frac{\partial \tau}{\partial \theta}, \qquad \theta \in \{c_1, d_1\}. \tag{29}$$

Gradient of τ w.r.t. c_1: Using $\partial w_1/\partial c_1 = -1$, $\partial w_2/\partial c_1 = 1$, and $\frac{\partial(1/k)}{\partial c_1} = -\frac{1}{k^2}\frac{\partial k}{\partial c_1}$:

$$\begin{aligned}\frac{\partial \tau}{\partial c_1} &= \frac{-1}{k_0} + w_1\left(\frac{1}{h_0\,k_0}\right) + \frac{1}{k_1} + w_2\left(-\frac{1}{h_1\,k_1}\right)\\ &= \frac{w_1 - h_0}{h_0\,k_0} + \frac{h_1 - w_2}{h_1\,k_1} = \frac{2h_1}{h_0\,k_0} - \frac{2h_0}{h_1\,k_1},\end{aligned} \tag{30}$$

where we used $w_1 - h_0 = 2h_1$ and $w_2 - h_1 = 2h_0$.

Gradient of τ w.r.t. d_1: Since w_1, w_2 do not depend on d_1:

$$\frac{\partial \tau}{\partial d_1} = w_1\left(-\frac{1}{k_0^2}\cdot\frac{1}{h_0}\right) + w_2\left(-\frac{1}{k_1^2}\cdot\left(-\frac{1}{h_1}\right)\right) = -\frac{w_1}{h_0\,k_0^2} + \frac{w_2}{h_1\,k_1^2}. \tag{31}$$

Therefore:

$$\frac{\partial m_1}{\partial c_1} = -\frac{m_1^2}{6}\left(\frac{2h_1}{h_0\,k_0} - \frac{2h_0}{h_1\,k_1}\right), \qquad \frac{\partial m_1}{\partial d_1} = -\frac{m_1^2}{6}\left(-\frac{w_1}{h_0\,k_0^2} + \frac{w_2}{h_1\,k_1^2}\right). \tag{32}$$

Step 5: Per-segment assembly. On each segment, D depends on c_1 and d_1 through (i) the local coordinate o, (ii) the interval length h_i (which scales the derivative terms), (iii) the values d_i, d_{i+1}, and (iv) the derivatives m_i, m_{i+1}.

Segment $i = 0$: $t \in [-1, c_1]$, with $d_0 = -1$, $d_1 = d_1$, $o = (t + 1)/h_0$.

Dependence of o on c_1: Since $h_0 = c_1 + 1$,

$$\frac{\partial o}{\partial c_1} = -\frac{t+1}{h_0^2} = -\frac{o}{h_0}.$$

The full expression $D = H_0(-1) + H_1h_0m_0 + H_2d_1 + H_3h_0m_1$ depends on c_1 through o, h_0, m_0, and m_1:

$$\begin{aligned}\left.\frac{\partial D}{\partial c_1}\right|_{i=0} &= \frac{\partial D}{\partial o}\cdot\frac{\partial o}{\partial c_1} + \frac{\partial}{\partial c_1}\Big[H_1\,h_0\,m_0 + H_3\,h_0\,m_1\Big]\\ &= \frac{\partial D}{\partial o}\left(-\frac{o}{h_0}\right) + (H_1\,m_0 + H_3\,m_1) + H_1\,h_0\,\frac{\partial m_0}{\partial c_1} + H_3\,h_0\,\frac{\partial m_1}{\partial c_1}.\end{aligned} \tag{33}$$

The term $(H_1m_0 + H_3m_1)$ arises from $\frac{\partial h_0}{\partial c_1} = 1$.

For d_1, noting that $d_{i+1} = d_1$ appears via H_2, and h_0, o do not depend on d_1:

$$\left.\frac{\partial D}{\partial d_1}\right|_{i=0} = H_2 + H_1\,h_0\,\frac{\partial m_0}{\partial d_1} + H_3\,h_0\,\frac{\partial m_1}{\partial d_1}. \tag{34}$$

Segment $i = 1$: $t \in [c_1, 1]$, with $d_1 = d_1$, $d_2 = 1$, $o = (t - c_1)/h_1$.

Dependence of o on c_1: Since $h_1 = 1 - c_1$,

$$\frac{\partial o}{\partial c_1} = \frac{-h_1 - (t - c_1)(-1)}{h_1^2} = \frac{-h_1 + (t - c_1)}{h_1^2} = \frac{t - c_1 - h_1}{h_1^2} = \frac{t-1}{h_1^2}.$$

The full expression $D = H_0d_1 + H_1h_1m_1 + H_2(1) + H_3h_1m_2$ depends on c_1 through o, h_1, m_1, and m_2:

$$\begin{aligned}\left.\frac{\partial D}{\partial c_1}\right|_{i=1} &= \frac{\partial D}{\partial o}\cdot\frac{t-1}{h_1^2} + \frac{\partial}{\partial c_1}\Big[H_1\,h_1\,m_1 + H_3\,h_1\,m_2\Big]\\ &= \frac{\partial D}{\partial o}\left(\frac{t-1}{h_1^2}\right) - (H_1\,m_1 + H_3\,m_2) + H_1\,h_1\,\frac{\partial m_1}{\partial c_1} + H_3\,h_1\,\frac{\partial m_2}{\partial c_1}.\end{aligned} \tag{35}$$

The term $-(H_1m_1 + H_3m_2)$ arises from $\frac{\partial h_1}{\partial c_1} = -1$.

For d_1, noting that $d_i = d_1$ appears via H_0:

$$\left.\frac{\partial D}{\partial d_1}\right|_{i=1} = H_0 + H_1\,h_1\,\frac{\partial m_1}{\partial d_1} + H_3\,h_1\,\frac{\partial m_2}{\partial d_1}. \tag{36}$$

Step 6: Chain rule to learnable parameters.

$$\frac{\partial \mathcal{L}}{\partial a} = \frac{\partial \mathcal{L}}{\partial D}\,\frac{\partial D}{\partial c_1}\,\cdot\, 2\sigma(a)\big(1-\sigma(a)\big), \tag{37}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial D}\,\frac{\partial D}{\partial d_1}\,\cdot\, 2\sigma(b)\big(1-\sigma(b)\big). \tag{38}$$

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly state our contributions and main claims, which are supported by theoretical analysis in Section 3 and experimental results in Section 4.

   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: We discuss the limitations of our work in Section 5

   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 include formal theoretical results (e.g., theorems or lemmas). We provide mathematical derivations to motivate our method in Section 3.

   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: All hyperparameters and implementation details are provided in the experimental section Section 4

   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 will release our code and data upon acceptance of the paper.

   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: All training and evaluation details are specified in Section 4 and Appendix B

   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: Each experiment was run only once due to the high computational cost of training 3D Gaussian Splatting.

   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: [Yes]

   Justification: Compute resources are detailed in Section 4

   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: We have reviewed and comply with 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 work focuses on improving 3D reconstruction quality and has no direct negative 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 risks.

    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 cite the original papers for all datasets used (e.g., Mip-NeRF 360 [1], and Tanks and temples [20]). All datasets are publicly available for research purposes.

    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: This 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: This paper does not involve crowdsourcing or 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: This paper does not involve 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: LLMs are not used as a component of our core methodology.

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.