2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

SVR-GS: Spatially Variant Regularization for Probabilistic Masks in 3D Gaussian Splatting

Ashkan Taghipour[1], Vahid Naghshin[2], Benjamin Southwell[2],
Farid Boussaid[1], Hamid Laga[3] and Mohammed Bennamoun[1]

***Abstract*— 3D Gaussian Splatting (3DGS) enables fast, high-quality novel view synthesis but relies on densification followed by pruning to optimize the number of Gaussians. Existing mask-based pruning, such as MaskGS, regularizes the *global* mean of the mask, which is misaligned with the *local* per-pixel (per-ray) reconstruction loss that determines image quality along individual camera rays. This paper introduces SVR-GS, a spatially variant regularizer that renders a per-pixel spatial mask from each Gaussian's effective contribution along the ray, thereby applying sparsity pressure where it matters: on low-importance Gaussians. We explore three spatial-mask aggregation strategies, implement them in CUDA, and conduct a gradient analysis to motivate our final design. Extensive experiments on *Tanks&Temples*, *Deep Blending*, and *Mip-NeRF360* datasets demonstrate that, on average across the three datasets, the proposed SVR-GS reduces the number of Gaussians by 1.79 × compared to MaskGS and 5.63 × compared to 3DGS, while incurring only 0.50 dB and 0.40 dB PSNR drops, respectively. These gains translate into significantly smaller, faster, and more memory-efficient models, making them well-suited for real-time applications such as robotics, AR/VR, and mobile perception. Additional materials are available on our project page: `https://ashkantaghipour.github.io/svrgs/`.**

I. Introduction

Novel view synthesis (NVS) aims to render realistic images of a 3D scene from camera poses that were never observed [1]–[5]. Robotic autonomy relies on accurate 3D reconstruction, and novel view synthesis (NVS) complements it as a predictive sensor by rendering expected views from candidate poses. These renderings expose visibility and occlusions, improving next-best-view selection and safer planning [6]. In practice, this supports navigation [7], mapping [8], manipulation [9], and active perception [10]. A major step forward in NVS was the introduction of Neural Radiance Fields (NeRF) [11], [12], which represent a scene as a continuous function queried by a multi-layer perceptron (MLP). While NeRFs yield photorealistic rendering, they are computationally intensive [13], [14], typically requiring $\sim$ 150–200 MLP evaluations per pixel (e.g., 64 coarse + 128 fine), making them unsuitable for real-time deployment in robotics [15]—particularly under the constraints of limited compute and memory on embedded platforms. 3D Gaussian Splatting (3DGS) replaces numerous per-sample MLP queries with an explicit set of anisotropic 3D Gaussians [16]. At render time, each 3D Gaussian is projected to a 2D elliptical Gaussian footprint (a "splat") and composited front-to-back with alpha blending [17]. The pipeline runs efficiently on GPUs by grouping splats into 16×16-pixel tiles and compositing them in shared memory, avoiding per-ray MLP evaluations [18], [19].

However, constructing 3DGS representations typically combines densification, parameter optimization, and pruning [20]–[22]. Densification adds new Gaussians to improve coverage and detail by splitting large Gaussians, cloning high-gradient Gaussians, and seeding under-covered regions [23], [24], often yielding scenes with millions of Gaussians. Pruning subsequently removes Gaussians with minimal impact on the photometric reconstruction of the training views, thereby reducing memory usage and accelerating training and inference-time rendering [25]–[27].

In this context, recent work has focused on reducing the Gaussian count while preserving image quality [28]–[30]. Building on this trend, Compact3DGS [31] learns a binary mask per-Gaussian that gates both opacity and scale. Using a straight-through estimator, it hard-masks tiny or transparent Gaussians whose contribution is visually negligible, reducing the number of low-contribution Gaussians during training. Rate–Distortion Optimization (RDO) [32] learns a per-Gaussian mask on opacity and scale. During training, the masks remain soft and dynamic: Gaussians may be temporarily suppressed and later revived. Hard removal is applied only after optimization.

LightGaussian [33] assigns an importance score to each Gaussian based on its contribution across training views, and prunes those with consistently low scores. EfficientGS [34] streamlines 3DGS by improving densification with aggregated pixel-gradient magnitudes, densifying only Gaussians that remain non-steady and thereby avoiding unnecessary splits. For pruning, it adopts an opacity-based criterion to remove faint Gaussians. Reduced 3DGS [35] detects regions where Gaussians are densely packed and prunes those that heavily overlap with their neighbors to avoid overpopulation. PUP–3DGS [36] estimates a sensitivity score from the reconstruction-error Hessian and prunes Gaussians that show high spatial uncertainty and low impact on image quality. Taming 3DGS [37] and RN-Gaussian [38] limit the cloning and splitting of Gaussians by considering their position and correlation with nearby Gaussians, thereby controlling growth and reducing redundancy.

MaskGS [39] models pruning as learning a probability of existence for each Gaussian. During training, it constructs

[1]The University of Western Australia, Australia. [2]Dolby Laboratories, Sydney, Australia. [3]Murdoch University, Australia. Emails: ashkan.taghipour@research.uwa.edu.au, {vahid.naghshin, benjamin.southwell}@dolby.com, farid.boussaid@uwa.edu.au, h.laga@murdoch.edu.au, mohammed.bennamoun@uwa.edu.au

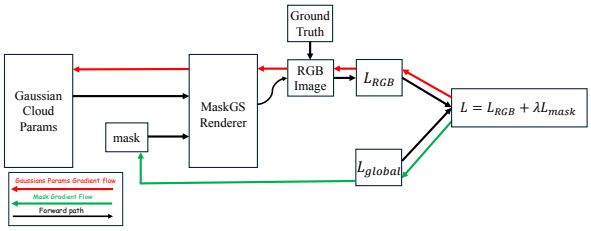

Fig. 1. MaskGS learns per-Gaussian existence probabilities and samples a binary mask to gate contributions while keeping Gaussian parameters trainable. However, its global mean-mask regularizer (Eq.3) enforces uniform sparsity, ignoring per-ray importance and motivating our spatially variant objective.

a binary mask by sampling this probability. This mask dictates whether a Gaussian contributes to the rendered image without destroying (zeroing) the Gaussian's parameters (e.g., position, covariance, opacity, color), which remain unchanged and trainable. The Gaussian's mask probabilities are regularized by adding the square of the mean Gaussian mask probability to the loss. Gradients flowing from the reconstruction loss to the Gaussian's that can improve the reconstruction quality result in the mask probabilities increasing (see Fig. 1). Pruning is applied at each densification step up to iteration 15k, and thereafter every 1,000 iterations until 30k; Gaussians are removed based on repeated stochastic sampling of their existence probabilities. As a result, the regularization term results in fewer Gaussians in the final optimized Gaussian cloud. Whilst achieving SOTA reconstruction quality on standard metrics and using fewer Gaussians than prior methods, MaskGS applies a uniform mask regularization that does not account for each Gaussian's contribution to the rendered images.

In this paper, we propose SVR-GS, a spatially variant regularization method for the probabilistic mask scheme proposed in MaskGS [39]. We do this by constructing a regularization function that considers the importance of each Gaussian in the RGB image by rendering a spatial mask image to regularize rather than simply taking the square of the mean probabilities. The spatial mask image penalizes each Gaussian according to its contribution, or more specifically, lack thereof, to pixels in the RGB image. This improves upon the MaskGS approach as we are now penalizing the existence of a Gaussian according to its contribution to the rendered images directly. The contributions of this paper can be summarized as:

- **Spatially variant mask objective.** A local, per-pixel regularizer on each Gaussian's *probability of existence* (mask probability), with weights derived from its visibility-weighted contribution to that pixel. This selectively penalizes Gaussians that contribute little to the rendered RGB image while preserving high-impact ones, aligning the pruning signal with the photometric reconstruction objective.
- **Design space analysis of spatial-mask renderers.** Three forward spatial mask rendering designs are implemented in CUDA and analyzed to explain their behavior dynamics; the ablation study reveals how design choices shape mask quality and reconstruction, motivating the final formulation.
- **Empirical effectiveness.** Comprehensive experiments on three real-world datasets show that, on average across the three datasets, our regularizer reduces the number of Gaussians by **1.79** × compared to MaskGS and **5.63** × compared to 3DGS, with PSNR drops of less than **0.5** dB (**0.50** and **0.40** dB), yielding substantially smaller models with negligible impact on reconstruction quality.

II. Proposed Method

A. Preliminaries: MaskGS

MaskGS [39] optimizes a mask probability for each Gaussian and uses a differentiable Gumbel–Softmax [40] during training. Masks do not modify a Gaussian's intrinsic attributes (e.g., opacity, shape, color); they simply decide whether it participates in compositing. Gaussians intersecting the ray for pixel x are composited in front-to-back order, with i=0 the nearest. Each Gaussian has spherical harmonics (SH) appearance coefficients $\mathbf{c}_i$; given the viewing direction $\mathbf{v}(x)$ at pixel x, its view-dependent color is $c_i(\mathbf{v}(x))$. The projected 2D elliptical footprint determines the per-pixel opacity $\alpha_i(x)$. The accumulated transmittance $T_i(x)$ is the fraction of light remaining before blending Gaussian i (with $T_0(x) = 1$). The transmittance and rendered color are then given by

$$T_{i+1}(x) \;=\; \big(1 - M_i\,\alpha_i(x)\big)\,T_i(x), \tag{1}$$

$$c(x) \;=\; \sum_{i=0}^{N-1} M_i\,\alpha_i(x)\,c_i\big(\mathbf{v}(x)\big)\,T_i(x). \tag{2}$$

Thus, M_i=1 includes Gaussian i (adds its color term and reduces transmittance); M_i=0 excludes it from Eq. 2 and leaves $T_{i+1}(x) = T_i(x)$ unchanged. So, the mask only decides whether a Gaussian participates in compositing; it does not delete or zero the Gaussian's parameters, which remain in the model and continue to be optimized. The overall objective combines a standard per-pixel reconstruction loss, $L_{\text{rgb}} = L_1 + (1 - \text{SSIM})$, where L_1 is the mean absolute error and SSIM is the structural similarity index measure, with a sparsity term on the average mask probability:

$$L \;=\; L_{\text{rgb}} \;+\; \underbrace{\lambda_m \left(\frac{1}{N} \sum_{i=0}^{N-1} M_i \right)^2}_{L_{\text{global}}}, \tag{3}$$

where N is the number of Gaussians and $\lambda_m > 0$ controls the regularization strength. Pruning is stochastic: at each pruning event, MaskGS [39] sample each Gaussian's mask 10 times and remove Gaussians that never activate; this is applied at every densification step and, thereafter, at fixed intervals of 1,000 iterations. A key limitation is that A single global average (second term of Eq. 3) treats all Gaussians equally, regardless of whether they are critical along a few rays or effectively invisible due to occlusion. So the penalty

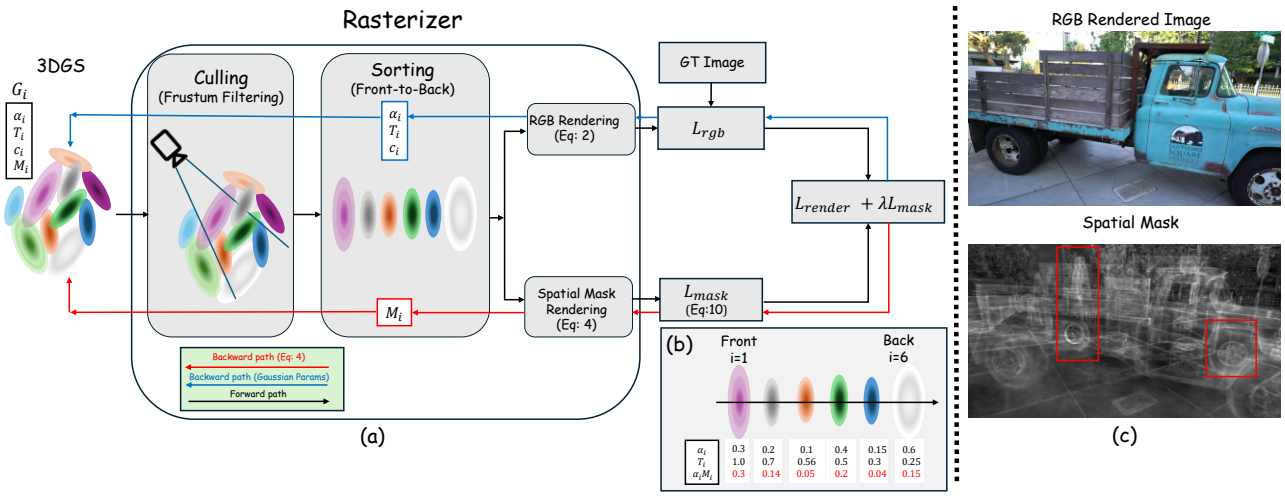

Fig. 2. **Pipeline overview.** (a) Starting from a set of 3D Gaussians G_i, Rasterizer cull splats outside the camera frustum and sort the remainder front-to-back. The renderer then branches into: (i) an RGB Rendering path that composites colors using Eq. (2); and (ii) a Spatial Mask rendering path that renders a Spatial Mask from the masks M_i (Eq. (4)). The training objective combines the image loss L_{rgb} with the mask loss as $L_{\text{render}} + \lambda L_{\text{mask}}$ (Eq. (11)). Blue and red arrows denote gradient flow from the RGB and mask losses, respectively; black arrows indicate the forward pass. (b) Front-to-back ordering example (i=1 is nearest); the table lists per-splat opacity α_i, transmittance T_i, and masked contribution $\alpha_i M_i$. (c) Example outputs: RGB rendering (top) and the Spatial Mask (bottom), with regions highlighted (red boxes) where redundancy is emphasized.

pushes down both indiscriminately. As a result, the optimizer may suppress high-impact Gaussians that support fine detail while under-penalizing the ones that have low contributions, leading to either quality loss or insufficient pruning.

B. Spatial Mask Rendering

Our goal is a local, per-pixel spatial mask $F(x)$ that highlights *low-importance* Gaussians along each camera ray, namely those that are low-opacity or occluded. We use the per-Gaussian importance signal $\alpha_i(x)\,T_i(x)$. This quantity is large for front-most, high-opacity Gaussians with high transmittance $T_i(x)$, and it is small for occluded or low-opacity Gaussians. We invert this importance to score low-importance and aggregate it along the ray:

$$F(x) \;=\; \frac{1}{\log\big(1+N(x)\big)} \sum_{i=0}^{N(x)-1} M_i \Big(1-\alpha_i(x)\,T_i(x)\Big), \qquad (4)$$

where $N(x)$ denotes the number of Gaussians intersecting the ray through pixel x, $M_i \in \{0,1\}$ is the sampled binary mask, and $T_i(x)$ is the transmittance prior to blending Gaussian i (updated via Eq. 1), with initialization $T_0(x) = 1$. The factor $\log\big(1+N(x)\big)$ normalizes the scale across rays of different lengths. Intuitively, $F(x)$ becomes large when the ray contains many *low-importance* Gaussians (good pruning targets) and remains small when a few foreground Gaussians explain most of the rendered color $c(x)$ at that pixel (see Fig. 3.

C. Backward: Gradient of the Mask Regularizer

For a given pixel x, we order the Gaussians along the ray by depth: $i = 0$ is nearest, then $i = 1,\ldots,N-1$. We differentiate F with respect to a fixed mask M_i. We use j as a running index in sums and k as a dummy index in products. For clarity of presentation, we omit the pixel index x.

Fig. 3. Spatial mask $F(x)$ rendered via Eq. (4) (left) alongside the RGB image (right). High intensities correspond to small per-Gaussian contributions $\alpha_i(x)T_i(x)$, highlighting occluded or faint structures.

We write the transmittance as

$$T_j \;=\; \prod_{k<j}\big(1-\alpha_k M_k\big), \qquad T_0 = 1. \qquad (5)$$

With this setup, $\partial F/\partial M_i$ has two parts: a direct effect on the i-th term and an indirect occlusion effect on later Gaussians.

Let $f_i = M_i\big(1-\alpha_i T_i\big)$, so $F = \frac{1}{\log(1+N)}\sum_i f_i$.

Self term (index i). Since T_i depends only on masks in front ($k < i$), it does not depend on M_i:

$$\frac{\partial f_i}{\partial M_i} \;=\; 1-\alpha_i T_i. \qquad (6)$$

Occlusion term (indices $j > i$). For $j > i$, $f_j = M_j(1-\alpha_j T_j)$ depends on M_i only through T_j. Using $T_j = \prod_{k<j}(1-\alpha_k M_k)$,

only the factor $(1-\alpha_i M_i)$ depends on M_i. Thus,

$$\frac{\partial T_j}{\partial M_i} = \left(\prod_{\substack{k<j \\ k\neq i}} (1-\alpha_k M_k)\right)(-\alpha_i) = -\alpha_i \frac{T_j}{1-\alpha_i M_i}. \quad (7)$$

By the chain rule,

$$\frac{\partial f_j}{\partial M_i} = \frac{\partial f_j}{\partial T_j}\frac{\partial T_j}{\partial M_i} = (-M_j\alpha_j)\left(-\alpha_i \frac{T_j}{1-\alpha_i M_i}\right) = M_j\,\alpha_j\,\alpha_i \frac{T_j}{1-\alpha_i M_i}. \quad (8)$$

Summing both parts and carrying the constant $1/\log(1+N)$ gives

$$\frac{\partial F}{\partial M_i} = \frac{1}{\log(1+N)}\left[(1-\alpha_i T_i) + \frac{\alpha_i}{1-\alpha_i M_i}\sum_{j>i} M_j\,\alpha_j\,T_j\right]. \quad (9)$$

Equation (9) decomposes the gradient into two interpretable components along the ray. The self term, $1-\alpha_i T_i$, the amount of contribution of each Gaussian: when a Gaussian is faint or occluded, $\alpha_i T_i$ is small, so $1-\alpha_i T_i$ is large. In that case, increasing its mask would raise F the most, meaning the spatial mask assigns a stronger penalty to such *low-importance* Gaussians. The occlusion term, $\frac{\alpha_i}{1-\alpha_i M_i}\sum_{j>i} M_j\alpha_j T_j$, captures the influence of i on all the Gausians behind it. If i has non-trivial opacity and there is a long, active tail of Gaussians deeper on the ray, this sum is large: keeping i occludes many others, so the gradient grows and discourages M_i. If small Gaussians lie behind, this occlusion term is small. Together, these effects form a simple behavior: weak or occluded Gaussians receive a stronger push toward smaller masks, while strong foreground Gaussians—especially when little lies behind—see a much milder signal and are preserved.

Beyond these per-Gaussian effects, the same decomposition predicts how the gradient scales with the overall density of active masks. The gradient becomes stronger when a larger share of masks are active. Let p denote the probability that a mask is "on". As p increases, more Gaussians participate along each ray and transmittance T_i becomes smaller for deeper indices, which enlarges the self term $1-\alpha_i T_i$, and the interaction term grows because the tail sum $\sum_{j>i} M_j\alpha_j T_j$ is larger. This yields larger $|\partial F/\partial M_i|$ in high-density settings than in sparse ones. Thus, the signal adapts to the *population* of active masks: it pushes harder when many masks are on (e.g., $p\approx 0.8$) and more gently when few are on (e.g., $p\approx 0.1$).

D. Spatial Mask Loss

Given the *spatial mask* from Eq. (4), let $x=(u,v)$ denote a pixel on the $H\times W$ image grid and write $F(x)\equiv F(u,v)$. We define the per-view mask loss as the mean energy of this spatial mask:

$$L_{\text{mask}} = \frac{1}{HW}\sum_{u=1}^{H}\sum_{v=1}^{W} F(u,v)^2 \quad (10)$$

The total objective combines RGB reconstruction loss with this regularizer:

$$L = L_{\text{rgb}} + \lambda_F\, L_{\text{mask}}, \quad (11)$$

where $\lambda_F \geq 0$ is a scalar hyperparameter that controls the strength of the spatial-mask penalty. For pruning, we follow the exact setup as MaskGS [39] explained in Section II-A

III. Experiments

A. Dataset and Metrics

We conduct experiments on three benchmark datasets. The Tanks&Temples dataset [41] provides outdoor scenes, while the Deep Blending dataset [42] focuses on indoor environments. In addition, we include Mip-NeRF360 [43], which contains both indoor and outdoor scenes. We evaluate performance using Peak Signal-to-Noise Ratio (PSNR) [44], which measures pixel-level differences; structural similarity (SSIM) [45], which reflects consistency of luminance, contrast, and structure; and learned perceptual image patch similarity (LPIPS) [46], which compares image features in a neural network space. We also report the number of Gaussians (#GS).

B. Compared Baselines

We compare our approach with four methods. The original 3DGS [16] serves as the primary reference. LightGaussian [33] uses score-based pruning followed by a standard recovery phase (continued optimization without densification or additional pruning), a practice we also adopt. Compact3DGS [31] prunes via a learned volume mask, and MaskGS [39] employs probabilistic masks with masked rasterization.

C. Implementation Details

We integrate our method into the MaskGS [39] rasterizer. Since the spatial-mask computation and its gradients are tightly coupled to the renderer's per-splat compositing, there is no practical way to implement them in Python; we therefore implement both parts inside the CUDA rasterizer. In the forward pass, the spatial mask F is computed alongside standard RGB compositing; in the backward pass, we compute the gradients in CUDA and make them available to PyTorch for backpropagation. For all methods (ours and baselines), each scene is trained for 30K steps, followed by a 5K step recovery phase that optimizes only the 3DGS parameters (no densification or pruning), as in LightGaussian [33]. All results in Table I use this 30k + 5k schedule.

D. Quantitative Results

Tanks&Temples (outdoor). Compared to MaskGS, our method reduces #GS by **2.2**× (from 0.590M to 0.272M); relative to 3DGS, the reduction is **6.7**× (from 1.825M to 0.272M). Quality changes are small: PSNR 23.25 vs. 23.75 (−0.50 dB), SSIM 0.829 vs. 0.847 (−0.018), LPIPS 0.221 vs. 0.178 (+0.043). In other words, the reduction in #GS is substantial, but the loss in image quality is comparatively small. Relative to the original 3DGS baseline, we achieve

TABLE I

QUANTITATIVE COMPARISON ON THREE DATASETS; BEST RESULTS ARE HIGHLIGHTED. #GS DENOTES MILLIONS OF GAUSSIANS; ↑ HIGHER IS BETTER, ↓ LOWER IS BETTER.

Dataset	Tanks&Temples				Deep Blending				Mip-NeRF360			
Metrics	PSNR↑	SSIM↑	LPIPS↓	#GS↓	PSNR↑	SSIM↑	LPIPS↓	#GS↓	PSNR↑	SSIM↑	LPIPS↓	#GS↓
3DGS (SIGGRAPH 23)	23.62	0.847	0.176	1.825	29.54	0.905	0.244	2.815	27.51	0.811	0.224	3.204
LightGaussian (NIPs 24)	23.76	0.843	0.187	0.626	29.55	0.902	0.250	0.959	27.50	0.809	0.232	1.084
Compact3DGS (CVPR 24)	23.68	0.849	0.179	0.960	29.63	0.903	0.245	1.310	27.34	0.812	0.233	1.533
MaskGS (CVPR 25)	23.75	0.847	0.178	0.590	29.71	0.905	0.246	0.694	27.49	0.811	0.228	1.205
Ours	23.25	0.829	0.221	0.272	29.46	0.904	0.261	0.255	26.75	0.800	0.255	0.866

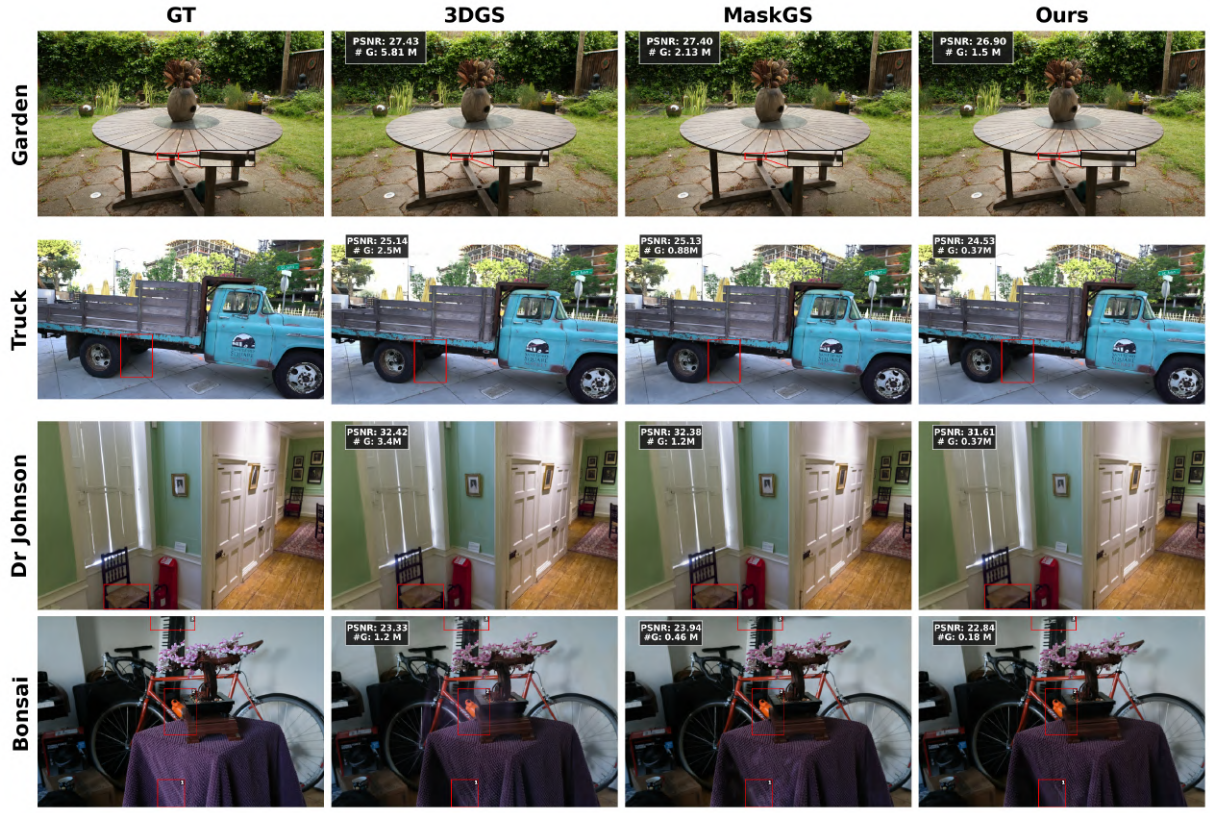

Fig. 4. Qualitative comparison of our method with 3DGS and MaskGS. For each scene, we report PSNR (dB, ↑) and #G (M, ↓), where #G denotes the number of Gaussians in millions. Regions with differences are marked in red boxes.

a **6.7** × reduction in #GS (from 1.825 M to 0.272 M). The qualitative results (Section III-E) provide visual context for these differences.

Deep Blending (indoor). We obtain a reduction in #GS of ≈ 2.7× fewer (from 0.694 M to 0.255 M) relative to MaskGS, with small changes in quality: PSNR 29.46 vs. 29.71 (−0.25 dB), SSIM 0.904 vs. 0.905 (−0.001), and LPIPS 0.261 vs. 0.246 (+0.015). The near-identical SSIM suggests structural details are preserved despite aggressive pruning. Compared to the original 3DGS baseline, the reduction is ≈ 11.0× fewer (from 2.815 M to 0.255 M). This reduction lowers both on-disk model size and GPU memory during rendering, and it reduces per-frame compositing cost. We use the same hyperparameters across scenes, and observe similar reductions without dataset-specific tuning.

Mip-NeRF360 (indoor/outdoor). On this mixed benchmark, #GS decreases by ≈ 1.4× fewer (from 1.205 M to 0.866 M) versus MaskGS. The quality gap is moderate: PSNR 26.75 vs. 27.49 (−0.74 dB), SSIM 0.800 vs. 0.811 (−0.011), and LPIPS 0.255 vs. 0.228 (+0.027). Relative to the original 3DGS baseline, the reduction is ≈ 3.7× fewer (from 3.204 M to 0.866 M). These outcomes are consistent with the challenge of *unbounded* outdoor views in Mip-NeRF360—scenes that cannot be enclosed within a small finite box around the cameras (e.g., with sky or distant geometry). We use the same hyperparameters across all datasets for fairness.

E. Qualitative Results

Qualitative comparisons. We present visual comparisons for novel-view synthesis in Fig. 4. In the top row (Garden),

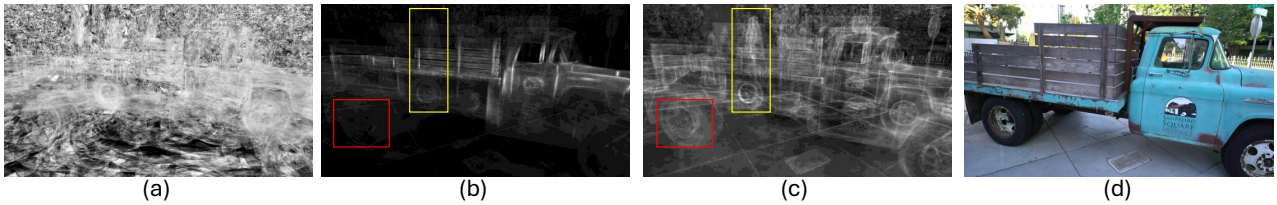

Fig. 5. **Effect of the forward function on the spatial mask.** We visualize the spatial mask $F(x)$ produced by three forward designs; brighter intensities indicate higher penalty / lower importance. (a) *Scenario A* (Eq. (13)): the inverse-importance weights $w_i = 1/(\alpha_i T_i + \varepsilon)$ explode where $\alpha_i T_i \ll 1$, causing saturation and broad regions to be marked as *low-importance*. (b) *Scenario B* (Eq. (16)): the cumulative-transmittance term $1 - T_i$ emphasizes occluded content (yellow box) but tends to over-penalize with depth and under-respond to faint foreground structure (red box). (c) *Proposed* spatial mask (Eq. (4)): the factor $1 - \alpha_i T_i$ highlights occluded areas (yellow) while also capturing non-occluded, low-opacity regions (red) with reduced background noise. (d) Reference RGB image.

the PSNR difference between our method and 3DGS is 0.53 dB; the only visible change is a slightly blurred edge on the table, which is hard to notice without zooming. In terms of #GS, our method uses 1.42× fewer Gaussians than MaskGS and 3.87× fewer than 3DGS. Overall, the quality difference is small compared with the substantial reduction in the number of Gaussians.

The second row of Fig. 4, shows a scene from the Tanks&Temples dataset. The PSNR gap between our method and 3DGS (which achieves the highest PSNR for this scene) is 0.61 dB. The main visual difference is that the partially occluded rear wheel of the truck appears slightly less sharp in our result compared to 3DGS and MaskGS. However, this sharper detail comes at a cost: our method uses 2.38× fewer Gaussians than MaskGS and 6.76× fewer than 3DGS. Thus, the loss in visual quality is minor relative to the substantial reduction in the number of Gaussians.

Third row of Fig. 4, shows a scene from the Deep Blending dataset (Dr Johnson). The PSNR gap between our method and MaskGS is 0.77 dB. As highlighted with red box in the rendered images, the only noticeable difference is a slight variation in the chair's texture color between the two methods. This accounts for the 0.77 dB difference, while no meaningful visual degradation—such as deformation, missing regions, or other artifacts—can be observed in the rendered images. Importantly, our method achieves this result with only 0.37M Gaussians, which is 3.24× fewer than MaskGS (1.2M) and 9.19× fewer than 3DGS (3.4M).

In the fourth row of Fig. 4, we present an extreme case where the PSNR difference between our method and MaskGS is 1.10 dB. As highlighted by the red boxes: in box 2, both our result and MaskGS render the region cleanly, whereas 3DGS shows slight blur; in box 3, none of the methods recovers the keyboard edge precisely; and in box 1, the only notable difference between our result and MaskGS is a subtle fade in the fabric texture. Despite these local deviations, our rendering uses 2.56× fewer Gaussians than MaskGS and 6.67× fewer than 3DGS, representing a substantial reduction in #GS for a comparatively small change in image quality.

TABLE II

ABLATION OF FORWARD DESIGNS FOR THE SPATIAL MASK ON TANKS&TEMPLES (#GS DENOTES MILLIONS OF GAUSSIANS)

Method	PSNR↑	SSIM↑	LPIPS↓	#GS↓
Scenario A	21.12	0.774	0.253	0.365
Scenario B	22.82	0.803	0.239	0.471
Ours	23.25	0.829	0.221	0.272

F. Effect of λ_F

Increasing λ_F scales the spatial-mask penalty in Eq. (11), emphasizing $L_{\text{mask}} = \frac{1}{HW}\sum_{u,v} F(u,v)^2$. This drives the gradients of *low-importance* Gaussians (small $\alpha_i T_i$) toward zero, inducing earlier and stronger pruning (as in Section. II-A).

We sweep $\lambda_F \in \{1, 1.25, 1.6, 2\} \times 10^{-4}$. As shown in Fig. 6, a larger λ_F consistently yields fewer Gaussians throughout training and a lower final count, with a corresponding PSNR drop. $\lambda_F = 1 \times 10^{-4}$ gives the best PSNR; $\lambda_F = 1.25 \times 10^{-4}$ is close (about 0.3–0.5 dB lower) while using $\sim$8–12% fewer Gaussians. Higher values ($1.6–2 \times 10^{-4}$) prune more but incur a $\sim$0.7–1.2 dB PSNR loss.

G. Ablations on the Forward Design

Designing the forward mask aggregation is critical because it determines the signal our loss will propagate. We did not assume the final form a priori; instead, we treated it as a hypothesis and tested two other alternatives that emphasize low-importance Gaussians. Each variant was implemented in CUDA for a fair comparison; see Fig. 5, and quantitative results are reported in Table II. We adopt Eq. (4) because it is stable, aligns with alpha compositing, and provides clean, importance-aware gradients. Below, we summarize the two alternatives and explain why they fall short.

1) Scenario A: Inverse-importance weighting: Here, we treat the *inverse* of per-Gaussian importance, $1/(\alpha_i T_i)$, as the low-importance signal, so Gaussians with small $\alpha_i T_i$ receive larger weight. We keep the transmittance:

$$T_0 = 1, \qquad T_{i+1} = (1 - \alpha_i M_i)\, T_i, \tag{12}$$

and define the weights and normalized aggregation

$$w_i = \frac{1}{\alpha_i T_i + \varepsilon}, \qquad F_A = \frac{\sum_i w_i M_i}{\sum_i w_i}. \tag{13}$$

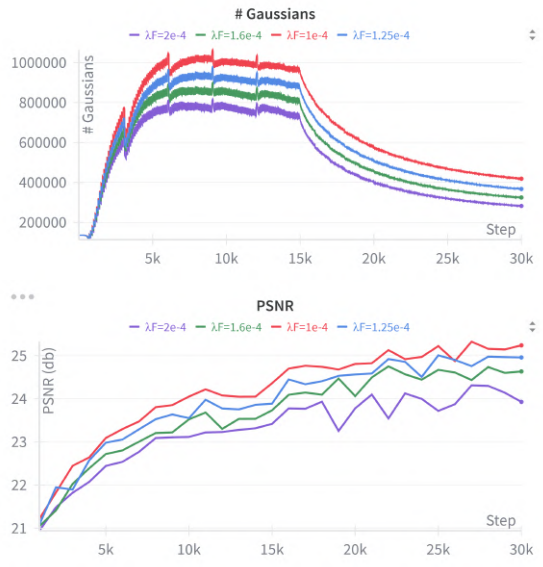

Fig. 6. Effect of λ_F on number of Gaussians (top) and quality (bottom). Larger λ_F prunes more and ends with fewer Gaussians, but PSNR drops.

For shorthand, let

$$S := \sum_k w_k, \qquad W_i^{>} := \sum_{j>i} w_j, \qquad F_i^{>} := \sum_{j>i} w_j M_j. \tag{14}$$

The gradient with respect to M_i can be written as

$$\begin{aligned} \frac{\partial F_A}{\partial M_i} &= \frac{1}{S}\left[w_i + \sum_{j>i}(M_j - F_A)\,\frac{\partial w_j}{\partial M_i}\right], \\ \frac{\partial w_j}{\partial M_i} &= \frac{\alpha_i \alpha_j T_j}{(1-\alpha_i M_i)\,(\alpha_j T_j + \varepsilon)^2}. \end{aligned} \tag{15}$$

Scenario A falls short because the inverse-importance weights $w_i = 1/(\alpha_i T_i + \varepsilon)$ explode when $\alpha_i T_i$ is tiny, saturating F_A and tagging broad regions as low-importance (Fig.5 (a)). As a result, the mask overreacts to near-zero contributions and background noise instead of reliably isolating truly low-important Gaussians.

2) Scenario B: Cumulative-transmittance masking: Here, we suppress each Gaussian by the cumulative occlusion in front of it. With the standard transmittance,

$$\begin{aligned} T_0 &= 1, & T_{i+1} &= (1-\alpha_i M_i)\,T_i, \\ f_i &:= M_i(1-T_i), & F_B &= \frac{1}{\log(1+N)}\sum_i f_i. \end{aligned} \tag{16}$$

The gradient w.r.t. M_i is

$$\frac{\partial F_B}{\partial M_i} = \frac{1}{\log(1+N)}\left[(1-T_i) + \frac{\alpha_i}{1-\alpha_i M_i}\sum_{j>i} M_j T_j\right]. \tag{17}$$

Scenario B falls short because the cumulative term $(1-T_i)$ depends only on occlusion, so the mask primarily tracks how far along the ray a Gaussian sits rather than its per-Gaussian importance. This depth bias over-penalizes long rays and occluded regions while under-responding to faint foreground structure (Fig. 5 (b)).

We implemented all three scenarios in CUDA. The spatial masks are shown in Fig.5. As seen in the Table II, Scenario A produces fewer Gaussians than Scenario B (#GS 0.365M vs. 0.471M) but suffers a 1.7 dB PSNR drop (21.12 vs. 22.82) because its inverse-importance weighting over-penalizes background Gaussians, saturating the mask (Fig. 5(a)). By contrast, the proposed forward achieves the highest pruning rate (fewest Gaussians; #GS 0.272M) while also delivering the best reconstruction quality (highest PSNR/SSIM and lowest LPIPS).

The detailed mathematical derivations of the backward functions for Scenario A and Scenario B are provided on our project page.

IV. CONCLUSIONS

This work introduces **SVR-GS**, a spatially variant regularizer for probabilistic mask pruning in 3D Gaussian Splatting. It forms a local per-pixel mask from each Gaussian's visibility-weighted contribution, aligning sparsity with the reconstruction loss. A CUDA ablation design motivated our mask aggregation. Integrated into masked rasterization, and averaged over three real-world datasets, SVR-GS prunes Gaussians by **1.79**× vs. MaskGS and **5.63**× vs. 3DGS, with PSNR drops of at most **0.5** dB (**0.50** and **0.40** dB). These reductions shrink disk and GPU memory footprints and accelerate inference, supporting real-time robotics use cases.

ACKNOWLEDGMENT

The authors gratefully acknowledge Dolby Laboratories for supporting this work, which was conducted during an internship at Dolby Laboratories in Sydney, and The University of Western Australia (UWA) for providing the PhD scholarship.

REFERENCES

[1] J. Held, R. Vandeghen, A. Deliege, A. Hamdi, S. Giancola, A. Cioppa, A. Vedaldi, B. Ghanem, A. Tagliasacchi, and M. Van Droogenbroeck, "Triangle splatting for real-time radiance field rendering," *arXiv preprint arXiv:2505.19175*, 2025.

[2] S. He, P. Ji, Y. Yang, C. Wang, J. Ji, Y. Wang, and H. Ding, "A survey on 3d gaussian splatting applications: Segmentation, editing, and generation," *arXiv preprint arXiv:2508.09977*, 2025.

[3] J. Held, R. Vandeghen, A. Hamdi, A. Deliege, A. Cioppa, S. Giancola, A. Vedaldi, B. Ghanem, and M. Van Droogenbroeck, "3d convex splatting: Radiance field rendering with 3d smooth convexes," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 21 360–21 369.

[4] N. von Lützow and M. Nießner, "Linprim: Linear primitives for differentiable volumetric rendering," *arXiv preprint arXiv:2501.16312*, 2025.

[5] H. Zeng, Y. Bai, and Y. Fu, "Arbitrary-scale 3d gaussian super-resolution," *arXiv preprint arXiv:2508.16467*, 2025.

[6] A. Zhou, M. J. Kim, L. Wang, P. Florence, and C. Finn, "Nerf in the palm of your hand: Corrective augmentation for robotics via novel-view synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 907–17 917.

[7] R. Jin, Y. Gao, Y. Wang, Y. Wu, H. Lu, C. Xu, and F. Gao, "Gs-planner: A gaussian-splatting-based planning framework for active high-fidelity reconstruction," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 11 202–11 209.

[8] K. Wu, K. Zhang, Z. Zhang, M. Tie, S. Yuan, J. Zhao, Z. Gan, and W. Ding, "Hgs-mapping: Online dense mapping using hybrid gaussian representation in urban scenes," *IEEE Robotics and Automation Letters*, vol. 9, no. 11, pp. 9573–9580, 2024.

[9] Y. Zheng, X. Chen, Y. Zheng, S. Gu, R. Yang, B. Jin, P. Li, C. Zhong, Z. Wang, L. Liu, C. Yang, D. Wang, Z. Chen, X. Long, and M. Wang, "Gaussiangrasper: 3d language gaussian splatting for open-vocabulary robotic grasping," *IEEE Robotics and Automation Letters*, vol. 9, no. 9, pp. 7827–7834, 2024.

[10] Q. Gu, Z. Lv, D. Frost, S. Green, J. Straub, and C. Sweeney, "Egolifter: Open-world 3d segmentation for egocentric perception," in *European Conference on Computer Vision*. Springer, 2024, pp. 382–400.

[11] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[12] C.-Y. Lin, C.-H. Wu, C.-H. Yeh, S.-H. Yen, C. Sun, and Y.-L. Liu, "Frugalnerf: Fast convergence for extreme few-shot novel view synthesis without learned priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2025, pp. 11 227–11 238.

[13] Y. Qi, L. Zhu, Y. Zhang, and J. Li, "E2nerf: Event enhanced neural radiance fields from blurry images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 13 254–13 264.

[14] Z. Qu, K. Xu, G. P. Hancke, and R. W. Lau, "Lush-nerf: Lighting up and sharpening nerfs for low-light scenes," *arXiv preprint arXiv:2411.06757*, 2024.

[15] Y. Li and D. Pathak, "Object-aware gaussian splatting for robotic manipulation," in *ICRA 2024 Workshop on 3D Visual Representations for Robot Manipulation*, 2024. [Online]. Available: https://openreview.net/forum?id=gdRI43hDgo

[16] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[17] S. Zhu, G. Wang, X. Kong, D. Kong, and H. Wang, "3d gaussian splatting in robotics: A survey," *arXiv preprint arXiv:2410.12262*, 2024.

[18] J. Mao, B. Li, B. Ivanovic, Y. Chen, Y. Wang, Y. You, C. Xiao, D. Xu, M. Pavone, and Y. Wang, "Dreamdrive: Generative 4d scene modeling from street view images," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 367–374.

[19] S. Sabour, L. Goli, G. Kopanas, M. Matthews, D. Lagun, L. Guibas, A. Jacobson, D. Fleet, and A. Tagliasacchi, "Spotlesssplats: Ignoring distractors in 3d gaussian splatting," *ACM Trans. Graph.*, vol. 44, no. 2, Apr. 2025. [Online]. Available: https://doi.org/10.1145/3727143

[20] G. Fang and B. Wang, "Mini-splatting: Representing scenes with a constrained number of gaussians," in *European Conference on Computer Vision*. Springer, 2024, pp. 165–181.

[21] J. Zhang, J. Jiang, Y. Chen, K. Jiang, and X. Liu, "Cob-gs: Clear object boundaries in 3dgs segmentation based on boundary-adaptive gaussian splitting," in *CVPR*, 2025.

[22] S. Shen, T. Shao, K. Zhou, C. Jiang, and Y. Yang, "Enliveninggs: Active locomotion of 3dgs," in *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025, pp. 896–905.

[23] S. Sun, M. Mielle, A. J. Lilienthal, and M. Magnusson, "High-fidelity slam using gaussian splatting with rendering-guided densification and regularized optimization," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 10 476–10 482.

[24] H. Xiang, X. Li, K. Cheng, X. Lai, W. Zhang, Z. Liao, L. Zeng, and X. Liu, "Gaussianroom: Improving 3d gaussian splatting with sdf guidance and monocular cues for indoor scene reconstruction," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 2686–2693.

[25] S. Niedermayr, J. Stumpfegger, and R. Westermann, "Compressed 3d gaussian splatting for accelerated novel view synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 10 349–10 358.

[26] Y. Wang, Z. Li, L. Guo, W. Yang, A. Kot, and B. Wen, "Contextgs: Compact 3d gaussian splatting with anchor level context model," *Advances in neural information processing systems*, vol. 37, pp. 51 532–51 551, 2024.

[27] W. Morgenstern, F. Barthel, A. Hilsmann, and P. Eisert, "Compact 3d scene representation via self-organizing gaussian grids," in *European Conference on Computer Vision*. Springer, 2024, pp. 18–34.

[28] M. Niemeyer, F. Manhardt, M.-J. Rakotosaona, M. Oechsle, D. Duckworth, R. Gosula, K. Tateno, J. Bates, D. Kaeser, and F. Tombari, "Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps," in *2025 International Conference on 3D Vision (3DV)*. IEEE, 2025, pp. 134–144.

[29] P. Papantonakis, G. Kopanas, B. Kerbl, A. Lanvin, and G. Drettakis, "Reducing the memory footprint of 3d gaussian splatting," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 7, no. 1, pp. 1–17, 2024.

[30] Z. Zhang, T. Song, Y. Lee, L. Yang, C. Peng, R. Chellappa, and D. Fan, "Lp-3dgs: Learning to prune 3d gaussian splatting," *Advances in Neural Information Processing Systems*, vol. 37, pp. 122 434–122 457, 2024.

[31] J. C. Lee, D. Rho, X. Sun, J. H. Ko, and E. Park, "Compact 3d gaussian representation for radiance field," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 719–21 728.

[32] H. Wang, H. Zhu, T. He, R. Feng, J. Deng, J. Bian, and Z. Chen, "End-to-end rate-distortion optimized 3d gaussian representation," in *European Conference on Computer Vision*. Springer, 2024, pp. 76–92.

[33] Z. Fan, K. Wang, K. Wen, Z. Zhu, D. Xu, Z. Wang *et al.*, "Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps," *Advances in neural information processing systems*, vol. 37, pp. 140 138–140 158, 2024.

[34] W. Liu, T. Guan, B. Zhu, L. Xu, Z. Song, D. Li, Y. Wang, and W. Yang, "Efficientgs: Streamlining gaussian splatting for large-scale high-resolution scene representation," *IEEE MultiMedia*, 2025.

[35] P. Papantonakis, G. Kopanas, B. Kerbl, A. Lanvin, and G. Drettakis, "Reducing the memory footprint of 3d gaussian splatting," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 7, no. 1, pp. 1–17, 2024.

[36] A. Hanson, A. Tu, V. Singla, M. Jayawardhana, M. Zwicker, and T. Goldstein, "Pup 3d-gs: Principled uncertainty pruning for 3d gaussian splatting," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 5949–5958.

[37] S. S. Mallick, R. Goel, B. Kerbl, M. Steinberger, F. V. Carrasco, and F. De La Torre, "Taming 3dgs: High-quality radiance fields with limited resources," in *SIGGRAPH Asia 2024 Conference Papers*, 2024, pp. 1–11.

[38] O. Mahmoud and M. Gendrin, "On reducing the number of gaussians for radiance field real-time rendering," in *2024 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON)*, 2024, pp. 259–264.

[39] Y. Liu, Z. Zhong, Y. Zhan, S. Xu, and X. Sun, "Maskgaussian: Adaptive 3d gaussian representation from probabilistic masks," in *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, June 2025, pp. 681–690.

[40] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[41] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: benchmarking large-scale scene reconstruction," *ACM Trans. Graph.*, vol. 36, no. 4, Jul. 2017. [Online]. Available: https://doi.org/10.1145/3072959.3073599

[42] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow, "Deep blending for free-viewpoint image-based rendering," *ACM Trans. Graph.*, vol. 37, no. 6, Dec. 2018. [Online]. Available: https://doi.org/10.1145/3272127.3275084

[43] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-nerf 360: Unbounded anti-aliased neural radiance fields," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 5470–5479.

[44] A. Horé and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 2366–2369.

[45] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[46] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:4766599