ConeGS: Error-Guided Densification Using Pixel Cones for Improved Reconstruction With Fewer Primitives

Bartłomiej Baranowski Stefano Esposito Patricia Gschoßmann
Anpei Chen† Andreas Geiger
University of Tübingen, Tübingen AI Center
baranowskibrt.github.io/conegs

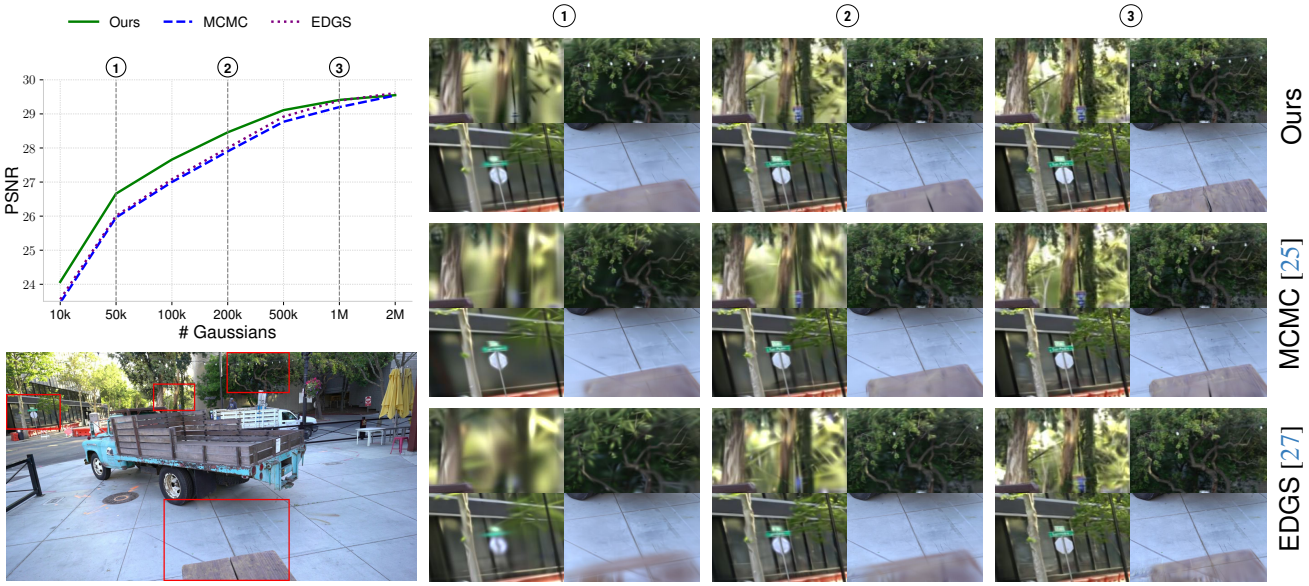


Figure 1. **ConeGS** replaces cloning-based densification with a novel method that generates pixel-cone-sized primitives in regions of high image-space error. By improving placement and removing reliance on existing scene structure thanks to a flexible iNGP-based exploration, it achieves higher reconstruction quality than baselines using the same number of primitives. Results are averaged over Mip-NeRF 360 [3] and OMMO [33] datasets, with a visual comparison on the `truck` scene from Tanks & Temples [26].

Abstract

3D Gaussian Splatting (3DGS) achieves state-of-the-art image quality and real-time performance in novel view synthesis but often suffers from a suboptimal spatial distribution of primitives. This issue stems from cloning-based densification, which propagates Gaussians along existing geometry, limiting exploration and requiring many primitives to adequately cover the scene. We present ConeGS, an image-space-informed densification framework that is independent of existing scene geometry state. ConeGS first creates a fast Instant Neural Graphics Primitives (iNGP) reconstruction as a geometric proxy to estimate per-pixel depth. During the subsequent 3DGS optimization, it identifies high-error pixels and inserts new Gaussians along the corresponding viewing cones at the predicted depth values, initializing their size according to the cone diameter. A pre-activation opacity penalty rapidly removes redundant Gaussians, while a primitive budgeting strategy controls the total number of primitives, either by a fixed budget or by adapting to scene complexity, ensuring high reconstruction quality. Experiments show that ConeGS consistently enhances reconstruction quality and rendering performance across Gaussian budgets, with especially strong gains under tight primitive constraints where efficient placement is crucial.

1. Introduction

Neural Radiance Fields (NeRF) [37] have significantly advanced novel view synthesis, achieving remarkable fidelity in scene reconstruction. However, representing scenes with

†Corresponding author.

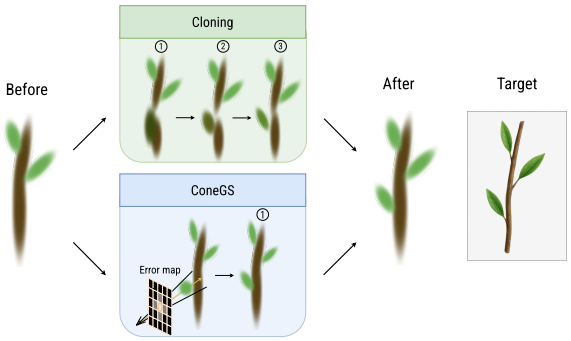


Figure 2. **Densification comparison.** Cloning-based methods are difficult to tune, and the resulting primitives may require many iterations to fit correctly into the scene. **ConeGS**, by contrast, places primitives precisely using the pixel viewing cone size, enabling faster scene integration without reliance on the existing geometry.

neural networks makes NeRF slow to train and render, though it provides smooth parameterization and flexibility to handle changes in scene structure. Recently, 3D Gaussian Splatting (3DGS) [24] has gained attention as a faster, more practical alternative to NeRF, explicitly modeling scenes with sets of 3D Gaussians to achieve interactive rendering speeds while maintaining competitive visual fidelity. However, 3DGS increases expressiveness through cloning and splitting, which offer limited exploration, rely on hard-to-define densification rules, and generate many unnecessary primitives. As a result, primitives often accumulate in suboptimal regions, leaving large parts of the scene underrepresented or mispredicted.

To address these issues, we propose **ConeGS**, which replaces cloning-based densification with a novel strategy that targets pixels exhibiting high photometric error. By sampling these pixels and using depth estimates from a fast Instant Neural Graphics Primitives (iNGP) [39] reconstruction, new Gaussians are placed precisely in regions where the current representation is insufficient. This targeted placement increases expressiveness in areas requiring higher primitive density, improving reconstruction quality while avoiding redundant primitives. To determine the size of new Gaussians, we draw inspiration from Mip-NeRF [2]. During densification, each Gaussian is initialized according to the size of the viewing cone of the pixel from which it is generated at the specified depth. Their initial size is thus defined directly by their image-space coverage, eliminating the need for local size analysis or adjustments to reconstructed regions. Figure 2 illustrates the effectiveness of the proposed approach. Combined with a pre-activation opacity penalty that quickly removes redundant Gaussians, this enables scene representation with fewer primitives while preserving high reconstruction quality. We further incorporate two primitive budgeting strategies to regulate the total number of primitives, either through a fixed budget or by adapting to scene complexity. ConeGS outperforms baseline methods across diverse datasets and a wide range of primitive budgets. The advantage is most pronounced under tight primitive budgets. At higher budgets, it matches the quality of cloning-based methods, where efficient primitive placement is less critical, while still rendering faster than the baselines. In summary, our contributions are:

- A densification strategy that places new Gaussians in regions of high photometric error in image space, guided by depth estimates from an iNGP-based geometric proxy.
- An approach that determines the size of new Gaussians from the viewing cones of the pixels from which they are generated.
- An improved opacity penalty that promptly removes low-opacity Gaussians, combined with a budgeting strategy that balances scene complexity and primitive count.

Finally, our method is also compatible with other 3DGS improvements, making it straightforward to integrate with existing approaches for greater efficiency, or with methods where cloning strategies are ambiguous or hard to formalize [20, 34, 49].

2. Related work

Neural Radiance Fields: NeRFs [37] represent scenes as continuous volumetric radiance fields, enabling high-quality novel view synthesis. This is achieved by parameterizing the scene with a neural network (typically an MLP), whose weights encode the scene globally. Despite producing photorealistic results, these methods rely on costly volumetric rendering and remain computationally inefficient. Extensions such as Mip-NeRF [2] and Mip-NeRF360 [3] reduce aliasing via conical frustum integration, while Zip-NeRF [4] improves view consistency with hierarchical sampling and multi-scale supervision. Hybrid approaches [7, 45, 46, 56] mitigate this by combining explicit data structures with compact neural representations, enabling faster optimization and real-time rendering. Instant Neural Graphics Primitives (iNGP) [39] further accelerate training through multi-resolution hash-grid encoding and shallow MLPs.

Primitive-based Differentiable Rendering: 3D Gaussian Splatting (3DGS) [24] has emerged as an efficient alternative to Neural Radiance Fields (NeRF) [37]. Rather than modeling the scene as a global volume, 3DGS represents it with local explicit 3D Gaussians and uses differentiable rasterization, resulting in significantly faster rendering. Its balance of fidelity and efficiency has attracted significant attention and spurred a wide range of follow-up research. Prior works have focused on tackling anti-aliasing [53, 57], reconstructing dynamic scenes [52, 54], enabling generative content creation [48, 66], reducing rendering arti-

facts [43], substituting alpha composition with volumetric rendering [35, 47], extracting geometry [16, 21, 58], level-of-detail reconstruction [44], frequency-based regularization [59, 60], and introducing new primitives or kernel functions [18, 20, 31, 49]. Recent efforts have also targeted reducing computational and memory costs, often through feature quantization or code-book encoding [10, 15, 34, 41], or scene simplification [63]. [11] reduces computation by lowering the number of primitives through an aggressive densification and pruning strategy, while [11, 12, 64] insert new Gaussians at the currently estimated depths using re-initialization. Unlike our method, this approach overwrites existing structures instead of adding new points, and further depends on the scene already being well reconstructed. [25] improves primitive distribution and exploration by incorporating positional errors and applying penalties to opacity and scaling. Closely related to our approach, several works focus on improving densification to reduce redundancy and better capture fine details. Strategies include refining cloning heuristics [5, 22, 25], per-Gaussian property- or saliency-based cues [36], geometry- and volume-aware criteria [1, 23, 65], addressing gradient collision [55], perceptual sensitivity [64], learnable schemes [32, 40], and based on Gaussian Processes [17]. Some works target densification in challenging settings [38], filling holes in the representation [9, 29], though typically adding only a few primitives. PixelSplat [6] models dense probability distributions for more robust Gaussian placement, influencing later approaches [8]. Recent work [27] suggests that densification may be unnecessary for high-quality reconstruction given strong initialization. Like our method, they start by estimating scene geometry, but rely only on correspondences from a pretrained dense matching network, without enhancing densification, and at higher GPU memory cost than our approach. Concurrent work [62], employs Gaussians with spatially varying texture colors, improving fine-detail reconstruction and reducing the number of primitives needed. Other methods use neural radiance fields for depth supervision [14, 28] or point cloud extraction [14, 42, 50] to initialize a scene, but not to improve densification directly. Concurrent work [13] applies NeRF for initialization and limited densification, constrained by existing Gaussian locations, and does not explore varying Gaussian sizes, which we find beneficial for reconstruction quality.

3. Preliminaries

3D Gaussian Splatting: 3DGS [24] represents a scene as an unordered set of 3D Gaussian primitives $\{\mathcal{G}_i | i = 1, \dots, M\}$. Each primitive $\mathcal{G}_i = (\mathbf{p}_i, \mathbf{s}_i, \mathbf{R}_i, o_i, \mathbf{c}_i)$ is defined by its position $\mathbf{p}_i \in \mathbb{R}^3$, scaling vector $\mathbf{s}_i \in \mathbb{R}^3$, rotation matrix $\mathbf{R}_i \in \mathbb{R}^{3\times3}$, opacity $o_i \in \mathbb{R}$, and view-dependent color $\mathbf{c}_i \in \mathbb{R}^3$. The color $\mathbf{c}_i$ is represented by spherical harmonics (SH) coefficients $\mathbf{k}_i \in \mathbb{R}^{3L}$, where L is the number of coefficients determined by the chosen SH order. The 3D covariance matrix is given by $\mathbf{\Sigma}_i = \mathbf{R}_i \mathbf{S}_i \mathbf{S}_i^T \mathbf{R}_i^T$, where $\mathbf{S}_i = \mathrm{diag}(\mathbf{s}_i)$ is the scaling matrix. The color $\hat{C}$ of a pixel is computed by α-blending over a set of N Gaussians, sorted by depth, whose projections overlap the pixel:

$$\hat{C} = \sum\nolimits_{i \in N} \mathbf{c}_i \alpha_i \prod\nolimits_{j=1}^{i-1} (1 - \alpha_j), \tag{1}$$

$$\alpha_i = o_i K(\mathbf{p}_c, \boldsymbol{\mu}_i^{\text{2D}}, \mathbf{\Sigma}_i^{\text{2D}}), \tag{2}$$

where α_i is the blending weight of the i-th Gaussian, $\mathbf{p}_c$ is the pixel center in image coordinates, $\boldsymbol{\mu}_i^{\text{2D}}$ and $\mathbf{\Sigma}_i^{\text{2D}}$ are the 2D projected mean and covariance of $\mathcal{G}_i$, and $K(\cdot)$ is a Gaussian filter response in screen space. The exact form of K depends on the chosen filter [24, 57, 67]. Gaussians are traditionally initialized from an SfM point cloud, with each component of $\mathbf{s}_i$ set equal to the mean Euclidean distance to the three nearest neighbors $\mathcal{N}_3(i)$ of Gaussian i:

$$\mathbf{s}_i = (s_i, s_i, s_i), \quad s_i = \frac{1}{3} \sum_{k \in \mathcal{N}_3(i)} \|\mathbf{p}_k - \mathbf{p}_i\| \quad . \tag{3}$$

During training, the Gaussian parameters are optimized with the loss:

$$\mathcal{L}_{\text{GS}} = (1 - \lambda)\, \text{MAE}(I, I^*) + \lambda\, \mathcal{L}_{D\text{-}SSIM}, \tag{4}$$

where $\lambda = 0.2$, MAE is the mean absolute error between the rendered image I and the ground-truth image I^*, and $\mathcal{L}_{D\text{-}SSIM} = 1 - \text{SSIM}(I, I^*)$ [51].

Neural Radiance Fields: NeRFs [37] model a scene as a continuous 3D field that maps a 3D location along a camera ray and the viewing direction of the corresponding pixel to a density $\sigma \in \mathbb{R}$ and color $\mathbf{c} \in \mathbb{R}^3$. A camera ray is parameterized as $\mathbf{r}(t) = \mathbf{p}_{\text{cam}} + t\mathbf{d}$, where $\mathbf{p}_{\text{cam}}$ is the camera position and $\mathbf{d}$ is a unit direction vector pointing toward the center of a pixel. Each ray is discretized into N intervals defined by distances $\{t_i, t_{i+1}\}_{i=1}^N$. For each sample position $\mathbf{r}(t_i)$ along the ray, the NeRF is queried to predict the sample's color $\mathbf{c}_i$ and density σ_i. Using volumetric rendering [37], the corresponding pixel color is approximated as:

$$\hat{C} = \sum\nolimits_{i=1}^{N} \mathbf{c}_i \alpha_i \prod\nolimits_{j=1}^{i-1} (1 - \alpha_j), \tag{5}$$

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i) \quad \text{with} \quad \delta_i = t_{i+1} - t_i. \tag{6}$$

Here, α_i is the opacity of the i-th sample, δ_i is the length of its ray segment, and the product term represents the transmittance $\tau_i = \prod_{j=1}^{i-1}(1 - \alpha_j)$.

Sampling only a single ray per pixel can lead to blur and aliasing. Mip-NeRF [2] addresses this by replacing the ray

with a cone that models the pixel footprint, i.e. the 3D volume a pixel covers in world space. The cone is divided into frustums, and integration is performed over these volumes rather than along a 1D line. The cone's radius $r_{\text{cone}}(t)$ defines the cross-section of the pixel cone at distance t and is computed from the directions of rays passing through the pixel and its neighbors:

$$r_{\text{cone}}(t) = t\,\frac{\|\mathbf{d}_x - \mathbf{d}\| + \|\mathbf{d}_y - \mathbf{d}\|}{2}, \tag{7}$$

where $\mathbf{d}$ is the direction of the ray through the center of the pixel, and $\mathbf{d}_x, \mathbf{d}_y$ are the directions of rays through the neighboring pixels in the x and y directions, respectively.

4. Method

This section outlines our ray-based densification approach for 3DGS. First, we explain how we use an iNGP model as a geometric proxy to initialize the 3D Gaussian scene (Section 4.1). Next, we detail our ray-based densification strategy, which uses the iNGP to place pixel-cone-sized Gaussians in high-error regions, along with associated optimization changes (Section 4.2). Finally, we provide additional implementation details in Section 4.3. An overview of the complete pipeline is shown in Figure 3.

4.1. Initialization

We use a trained iNGP model [39] as a geometric proxy to initialize the 3DGS scene and guide densification. Trained briefly on input images, it provides accurate depth estimates, that position both the initial Gaussian primitives and those added later during densification, with minimal impact on training time. Additionally, the depths can be evaluated on the fly during optimization, reducing both memory usage and computation compared to precomputing all depth maps. We initialize the scene with $\mathcal{P}_{\text{init}}$ Gaussians, set to one million as in [42], or fewer if a smaller budget is specified (see Section 4.2). To construct this set, we uniformly sample $\mathcal{P}_{\text{init}}$ image-pixel pairs (I, u, v) from the training set pixel domain $\mathcal{I}_{\text{train}}$. Each sampled image-pixel pair defines exactly one Gaussian in the initialized scene. For each sample, we define its associated camera ray

$$\mathbf{r}_I(u, v, t) = \mathbf{p}_I + t\,\mathbf{d}_I(u, v), \tag{8}$$

where I is an image index, (u, v) are pixel coordinates, $\mathbf{p}_I \in \mathbb{R}^3$ is the camera center, and $\mathbf{d}_I(u, v) \in \mathbb{R}^3$ is the normalized ray direction. We query the iNGP along this ray to obtain discrete transmittance values $\{\tau_k\}$, from which the median depth t_{med} is computed as:

$$t_{\text{med}} = t_k \quad \text{where} \quad \tau_{k-1} > 0.5 \geq \tau_k. \tag{9}$$

The center of the j-th Gaussian primitive is then set to:

$$\mathbf{p}_j = \mathbf{p}_I + t_{\text{med}}\,\mathbf{d}_I(u, v), \tag{10}$$

yielding the set of initial centers $\{\mathbf{p}_j\}_{j \in \mathcal{I}_{\text{sample}}}$ with $\mathcal{I}_{\text{sample}} \subset \mathcal{I}_{\text{train}}$. The scale $\mathbf{s}_j$ is initialized isotropically using the average distance to the three nearest neighbors $\mathcal{N}_3(j)$, following Eq. (3). The rotation is set to identity $\mathbf{R}_j = \mathbf{I}$, the opacity to $o_j = 0.1$, and the SH coefficients to:

$$\mathbf{k}_j = (\mathbf{k}_{1:3,j}, \mathbf{k}_{4:L,j}), \quad \mathbf{k}_{1:3,j} = \mathbf{c}_{j,0}, \quad \mathbf{k}_{4:L,j} = \mathbf{0}, \tag{11}$$

where $\mathbf{c}_{j,0}$ is the RGB color for the sampled pixel (I, u, v) rendered with iNGP using a zeroed-out view direction. Although our densification strategy can achieve high-quality reconstructions without scene initialization, we retain this step to ensure consistently strong performance across all metrics (see Section 5.2).

4.2. Optimization

We fully replace the standard 3DGS cloning-based densification with an error-guided strategy that adapts the iNGP-based ray-depth rendering procedure from Section 4.1 to position new Gaussian primitives. Below, we outline the sampling, scaling, budgeting, and pruning stages of our densification pipeline.

Error-Weighted Gaussian Densification: To limit the number of primitives while targeting poorly reconstructed regions, we add new Gaussians at pixels with high photometric error. At iteration j, we render an image I_j and compute the per-pixel absolute error (L_1 loss) $E(\mathbf{p}) = |I_j(\mathbf{p}) - I^*(\mathbf{p})|$ with respect to the ground-truth image I^*. We then sample N_{sample} pixels without replacement according to a multinomial distribution $\mathcal{M}$ with probabilities proportional to the normalized error map:

$$\{\mathbf{p}_s\}_{s=1}^{N_{\text{sample}}} \sim \mathcal{M}\left(N_{\text{sample}}, \frac{E(\mathbf{p})}{\sum_{\mathbf{p}' \in I_j} E(\mathbf{p}')}\right), \tag{12}$$

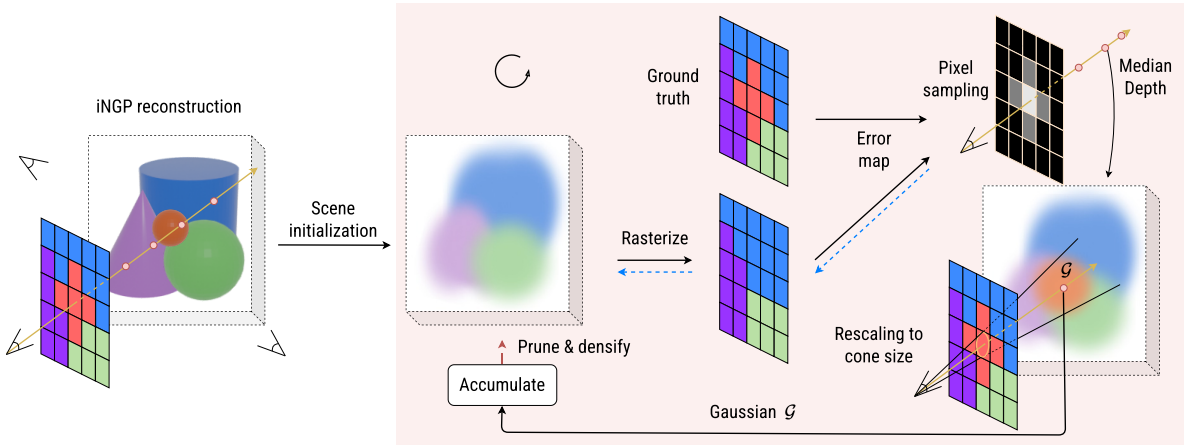
While L_1 loss does not always indicate possible improvements and can also arise from noise or difficult-to-optimize reflections, we found it to be a reliable indication of lacking expressiveness, especially at low primitive budgets. For each sampled pixel $\mathbf{p}_s$, a new Gaussian $\mathcal{G}_s$ is created, with its center placed along the corresponding ray at the median depth $t_{\text{med},s}$ given by the iNGP. Newly spawned Gaussians are appended to an accumulation set:

$$\mathcal{G}_{\text{accum}} \leftarrow \mathcal{G}_{\text{accum}} \cup \{\mathcal{G}_s\}_{s=1}^{N_{\text{sample}}}. \tag{13}$$

Every 100 iterations, low-opacity Gaussians in the scene $\mathcal{G}_{\text{scene}}$ are pruned, and Gaussians in $\mathcal{G}_{\text{accum}}$ are merged into the scene:

$$\mathcal{G}_{\text{scene}} \leftarrow \mathcal{G}_{\text{scene}} \cup \mathcal{G}_{\text{accum}}, \quad \mathcal{G}_{\text{accum}} \leftarrow \emptyset. \tag{14}$$

The newly inserted Gaussians are then jointly optimized with existing primitives. Unlike cloning-based methods,



(a) Initialization (Section 4.1)

(b) Optimization (Section 4.2)

Figure 3. **Overview of the ConeGS pipeline.** (a) First, an iNGP reconstruction is obtained to serve as a geometric proxy for object surfaces, guiding the placement of Gaussians both during scene initialization and throughout the 3DGS optimization process. (b) During 3DGS optimization, ConeGS performs error-guided densification by sampling a subset of pixels with high L_1 error. For each sampled pixel, a new Gaussian $\mathcal{G}$ is created along the pixel's viewing cone at the depth estimated by iNGP and scaled to match the cone's size. New Gaussians are accumulated and, every 100 iterations, inserted into the scene after pruning those with low opacity. Blue arrows indicate gradient updates to Gaussian parameters, and the red arrow marks scene updates.

which constrain new primitives to the vicinity of existing ones and thus hinder exploration of unseen regions, our approach places Gaussians directly in high-error areas, enabling effective scene coverage even far from existing geometry. Moreover, we preserve the integrity of well-reconstructed areas, since new Gaussians are added on top of the existing structure rather than created through splitting or cloning. Although occasional iNGP depth inaccuracies may introduce misplaced Gaussians, diverse viewpoint coverage ensures that inconsistent ones are quickly corrected or pruned, while multiview-consistent ones are retained. The full densification pipeline is shown in Figure 4.

Pixel-Footprint-Aligned Scaling: Selecting an appropriate scale for newly added Gaussians during densification is crucial. If primitives are too large, they may obscure fine details and be pruned prematurely, whereas overly small ones contribute little to the rendered image, yielding weak gradients and slowing convergence. Although k-NN–based scaling is effective for initialization, recomputing nearest-neighbor distances at every densification step is computationally expensive and sensitive to outliers. Large distances can produce inflated scales, causing new Gaussians to overlap well-reconstructed regions and hinder further optimization (see Section 5.2). To avoid these issues, we set the initial scale of each newly added Gaussian directly from the pixel footprint at the median depth $t_{\text{med},i}$ along its corresponding camera ray, as defined in Eq. 7:

$$\mathbf{s}_i = \lambda_{\text{scale}}\, r_{\text{cone}}(t_{\text{med},i})\,(1,1,1), \tag{15}$$

where $\lambda_{\text{scale}} = 2$ converts the cone radius $r_{\text{cone}}(t_{\text{med},i})$ to the diameter of its cross-section. This ensures that, from the spawning viewpoint, the Gaussian's projection onto the image plane approximately matches the pixel width, independent of scene depth. The assigned scale is only an initial value, with subsequent optimization steps jointly updating all primitives to allow newly added Gaussians to adjust to the existing scene. Our pixel-aligned, depth-aware scaling provides three key benefits: (1) it is independent of the current primitive distribution, avoiding the structural biases of cloning-based methods that replicate and reinforce local geometry, (2) pixel size allows Gaussians to contribute to optimization immediately and efficiently fit fine details while minimizing overlap with existing structure, and (3) its isotropic shape promotes stable multi-view integration, without shapes that the cloned Gaussians inherit.

Primitive Budgeting: We consider two budgeting strategies for controlling the number of Gaussians in the scene. The first enforces a hard upper bound, as in [25], ensuring that densification never exceeds the prescribed budget. This constraint regulates memory and computation while preventing uncontrolled growth of the primitive set. At each densification step, we set the number of sampled pixels N_{sample} so that newly added Gaussians replace those pruned, avoiding excess primitives that would otherwise be discarded under the budget. This is computed as:

$$N_{\text{sample}} = \frac{\max(0.2N_{\text{GS}},\ 1.2N_{\text{last}})}{100}, \tag{16}$$

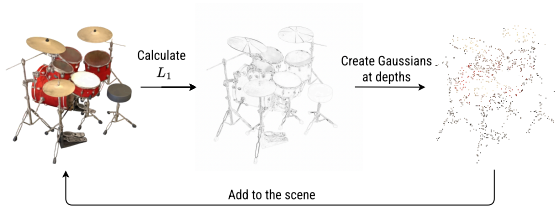


Figure 4. **Densification overview.** Illustration of the proposed error-guided strategy. We render an image with 3DGS, compute the per-pixel L_1 error, sample pixels proportionally to their error magnitude, and place new Gaussians at the iNGP-predicted depth along the corresponding viewing rays.

where N_{GS} is the current total number of Gaussians, and N_{last} is the number of primitives inserted in the previous densification step, and the division by 100 reflects the densification interval. This formulation keeps N_{GS} close to the budget limit even under aggressive pruning, maintaining consistent scene coverage throughout optimization.

The second strategy adapts the number of primitives to the scene's complexity, enabling controlled growth without imposing a fixed upper bound:

$$N_{\text{sample}} = \frac{\beta N_{\text{GS}}}{100}. \tag{17}$$

Here, β controls the growth rate of the primitive set. Smaller values balance the number of Gaussians added with those pruned, maintaining a relatively stable primitive count, whereas larger values yield higher primitive counts, increasing geometric detail at the cost of memory and computing power. With scene initialization at 1M primitives and the application of the opacity penalty, we balance Gaussians added and pruned, unlike [5], which requires a predefined upper limit on primitives.

Opacity-Regularized Pruning: Following [24, 25], Gaussians with opacity below 0.005 are pruned every 100 iterations to remove primitives with negligible contribution to the rendered image. Earlier work has promoted sparsity through different strategies: periodically resetting opacities [24], which can destabilize training [5], reducing opacities by a constant amount after each densification [5], or introducing a post-activation opacity penalty $\mathcal{L}_{\mathbf{o}}^{\text{post}} = \|\sigma(\mathbf{o}_{\text{pre}})\|_1$ [25], where $\mathbf{o}_{\text{pre}}$ denotes the opacity logits before the sigmoid and σ is the sigmoid function. This applies the strongest constraint around 0.5 and only a weak penalty near the pruning threshold. In contrast, we employ a pre-activation opacity penalty $\mathcal{L}_{\mathbf{o}}^{\text{pre}} = \|\mathbf{o}_{\text{pre}}\|_1$. It provides a steady constraint across the full opacity range, including very low values, gradually reducing under-contributing primitives. The penalty acts throughout training, and our densification strategy can freely add new primitives, allowing any structure lost through pruning to be recovered more easily than with cloning-based approaches. In all experiments, this loss is scaled by $\lambda_{\mathbf{o}} = 0.0002$.

4.3. Implementation Details

For the iNGP model, we use the proposal-based implementation from NerfAcc [30], trained for 20k iterations with the original setup and architecture. Gaussian optimization runs for 30k iterations, with our densification active for the first 25k. Unlike 3DGS, all SH components are optimized from the start, enabled by the stable initialization that removes the need for gradual SH introduction.

5. Evaluation

Dataset and Metrics: We evaluate our method on publicly available scenes from Mip-NeRF360 [3] and OMMO dataset [33], with `01` scene from OMMO resized to have 1600 pixels width. Following [24, 25], we also include the `train` and `truck` scene from Tanks & Temples [26], as well as `Dr Johnson` and `playroom` from the Deep-Blending [19] dataset. We report PSNR, SSIM [51], and LPIPS [61], with rendering speeds averaged across the full test set. All FPS measurements were recorded on an NVIDIA RTX 2080 Ti, whereas training speeds are reported on an NVIDIA A100, since EDGS requires more memory. These training speeds do not include later-added speed improvements [36].

Baselines: We primarily compare our method against 3DGS [24], it's extension with iNGP point cloud initialization [14], MCMC [25] using different initialization types (random, SfM, iNGP point clouds), as well as, GaussianPro [9], Perceptual-GS [64], EDGS (with densification) [27], with the densification stopped for all of them if the primitive budget is reached. If the number of primitives at initialization would be higher than the specified budget, the number of primitives is sampled uniformly to fit below it. In the random initialization settings we follow the process described in MCMC [25]. We additionally test on Mini-Splatting2 [11] by matching their final number of primitives instead of a specific budget, due to their method relying on generating a high number of initial Gaussians.

5.1. Results

We observe improvements over the baselines across a wide range of specified budgets in Table 1, with plots comparing the most important methods on the budget and no-budget scenario in Figure 6. For a lower limit on Gaussians, we outperform the benchmarks across all datasets and metrics, while providing a competitive reconstruction quality compared to the best performing baselines on the high budget scenarios. In Table 2 we additionally show that even on a high number of primitives of 1M and including the iNGP training, our method provides competitive speed to other

	Mip-NeRF360 [3]			OMMO [33]			Tanks & Temples [26]			DeepBlending [19]		
	PSNR ↑	**SSIM ↑**	**LPIPS ↓**	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**
Number of Gaussians limited to 100k												
3DGS [24] (SfM init.)	23.61	0.693	0.413	26.45	0.820	0.296	22.38	0.774	0.333	24.65	0.827	0.412
Foroutan et al. [14]†	26.64	0.781	0.318	26.89	0.829	0.276	22.48	0.766	0.341	25.31	0.822	0.418
MCMC [25] (rand. init.)	25.72	0.730	0.369	25.92	0.808	0.313	21.45	0.750	0.365	27.94	0.859	0.369
MCMC [25] (SfM init.)	27.06	0.800	0.303	27.01	0.841	0.266	22.50	0.780	0.332	28.94	0.876	0.333
MCMC [25] (iNGP init.)	27.35	0.797	0.299	26.95	0.837	0.265	22.69	0.775	0.326	29.02	0.872	0.337
GaussianPro [9]	25.57	0.766	0.338	26.14	0.822	0.289	20.59	0.757	0.348	28.15	0.870	0.342
Perceptual-GS [64]	25.66	0.774	0.320	26.13	0.819	0.292	20.76	0.759	0.347	28.32	0.874	0.338
EDGS [27]	27.09	0.798	0.296	26.99	0.838	0.261	22.32	0.777	0.324	28.43	0.872	0.337
Ours	27.74	0.809	0.285	27.59	0.852	0.243	23.12	0.791	0.310	29.44	0.880	0.328
Number of Gaussians limited to 500k												
3DGS [24] (SfM init.)	28.22	0.821	0.260	28.96	0.883	0.196	23.54	0.816	0.265	29.25	0.882	0.302
Foroutan et al. [14]†	28.88	0.862	0.204	28.80	0.884	0.185	23.52	0.816	0.257	29.61	0.886	0.297
MCMC [25] (rand. init.)	28.38	0.844	0.237	28.23	0.874	0.212	22.96	0.808	0.279	28.84	0.875	0.315
MCMC [25] (SfM init.)	28.82	0.861	0.214	28.72	0.885	0.194	23.68	0.825	0.259	29.44	0.887	0.308
MCMC [25] (iNGP init.)	29.02	0.867	0.198	28.75	0.885	0.186	23.57	0.823	0.243	29.61	0.885	0.288
GaussianPro [9]	28.02	0.827	0.253	28.32	0.876	0.206	22.31	0.802	0.286	29.33	0.886	0.301
Perceptual-GS [64]	28.66	0.856	0.211	28.42	0.876	0.203	22.77	0.813	0.267	29.44	0.888	0.296
EDGS [27]	28.82	0.865	0.193	29.01	0.893	0.168	23.47	0.831	0.229	29.33	0.889	0.286
Ours	29.08	0.870	0.190	29.14	0.892	0.170	23.69	0.829	0.229	29.86	0.891	0.285
Number of Gaussians bounded by Mini-Splatting2 [11]												
Mini-Splatting2 [11]	28.89	0.875	0.183	28.06	0.875	0.198	22.79	0.823	0.239	29.99	0.898	0.279
Ours	29.26	0.875	0.179	28.90	0.887	0.179	23.66	0.829	0.231	29.82	0.891	0.280

† Original code was not publicly available. Our implementation uses iNGP initialization and does not include the additional depth-based loss.

Table 1. **Quantitative results** with a Gaussian number limit of 100k and 500k. Mini-Splatting2 [11] does not support constraining the number of Gaussians during reconstruction, so we match its Gaussian count. We highlight the best, second best and third best results among methods with the same Gaussian counts. Per-scene metrics for selected methods are provided in the supplementary material.

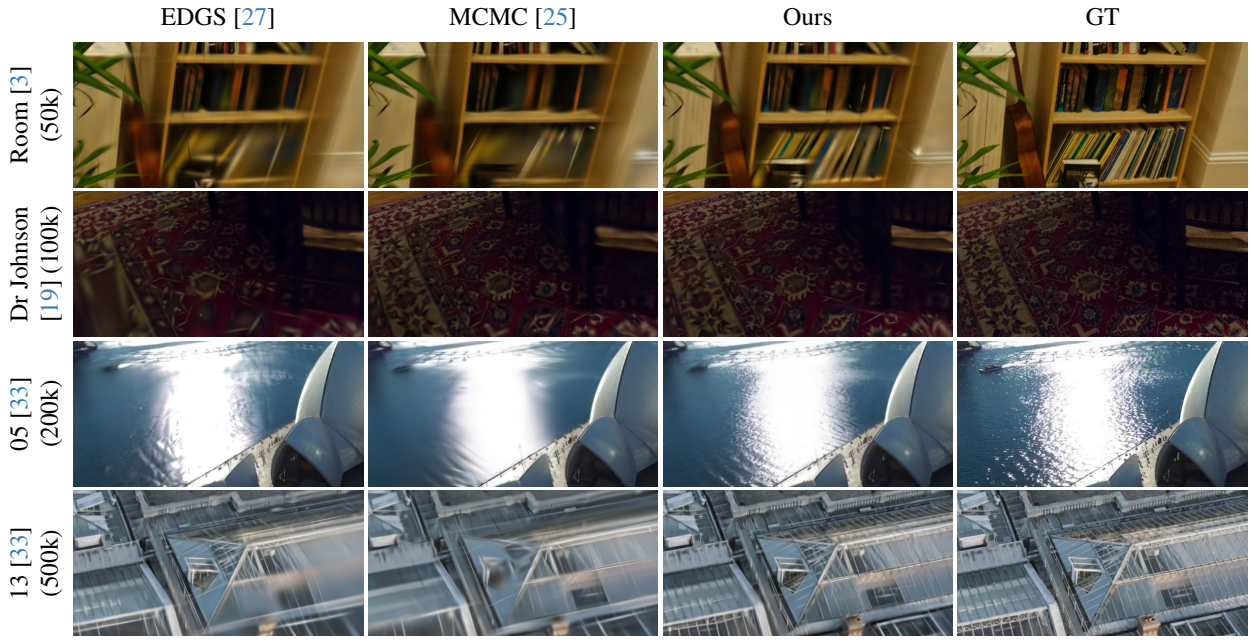


Figure 5. **Qualitative results** comparing our method with MCMC [25] (with SfM point cloud initialization) and EDGS [27] on the Mip-NeRF 360 [3], OMMO [33], and DeepBlending [19] datasets, with varying Gaussian budgets (given in parentheses).

methods. The qualitative results in Figure 5 show significant improvement using a wide range of primitive budgets, demonstrating that for the same limit of primitives our method is able to produce a much better reconstruction, especially in areas that are challenging to properly capture on a low budget, such as isolated or high frequency structures. We provide additional qualitative and quantitative results, along with further scene analysis, in the appendix.

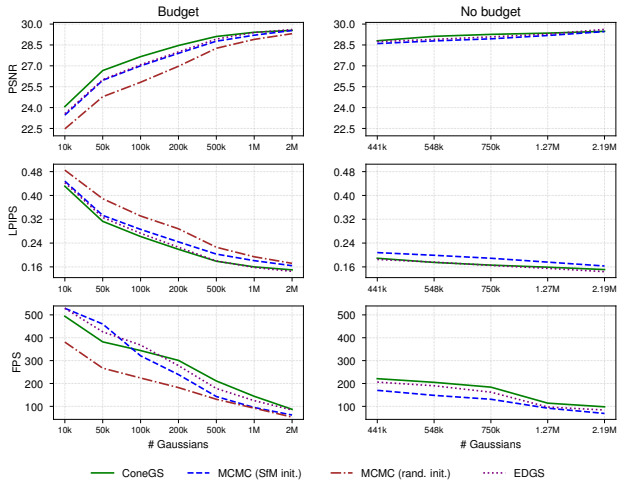


Figure 6. **PSNR, LPIPS and FPS plots** with (left) and without (right) a primitive budget, where counts correspond to β values. Averaged across Mip-NeRF360 [3] and OMMO [33]. Numerical results are provided in the appendix.

	Ours 10k iters	Ours 20k iters	Ours 40k iters	EDGS [27]	3DGS [24] (SfM init.)	MCMC [25] (rand. init.)	MCMC [25] (SfM init.)
PSNR ↑	29.33	29.37	29.37	29.18	28.71	28.98	29.23
SSIM ↑	0.877	0.880	0.881	0.877	0.846	0.865	0.875
LPIPS ↓	0.171	0.168	0.166	0.168	0.222	0.204	0.190
FPS ↑	134	137	137	131	112	92	95
3DGS time ↓	20.7	20.5	20.5	19.4	22.6	26.3	25.1
Init. time ↓	1.6	3.1	6.1	2.3	-	-	-
Overall time ↓	22.3	23.6	26.6	21.7	22.6	26.3	25.1

Table 2. **Quantitative results** averaged over Mip-NeRF360 [3] showing reconstruction timings for 3DGS [24], MCMC [25], EDGS [27], and our method (with varying iNGP durations), capped at 1M Gaussians. 3DGS time reports target scene optimization and densification, while init. time shows iNGP reconstruction for our method and initial matching for EDGS [27].

5.2. Ablations

We analyze each component's impact through ablation studies in Table 3. **(a), (b)** Longer iNGP reconstruction only slightly improves reconstruction quality. **(c)** continuing training the iNGP model also during optimization, **(d)** initializing Gaussians with the ground truth pixel color, or **(e)** predicting spherical harmonics with iNGP, leads to marginally worse results. Demonstrating the strength of our densification method, using **(f)** pixel-cone-sized primitives during initialization, or even not using any initialization **(g)**, results in worse PSNR but maintains low LPIPS and considerably improves rendering speed, thanks to less overlap between primitives, reducing blending. **(h)** Sampling pixels uniformly instead of guiding Gaussian creation using the L_1 loss from the training set produces lower reconstruction quality, although due to uniform sampling in image space still focusing more on parts of the scene seen most across views, the drop is not drastic. Densifying with

Ablation	**PSNR** ↑	**SSIM** ↑	**LPIPS** ↓	**FPS** ↑
Ours	27.74	0.810	0.285	328
(a) 10k iNGP iter.	27.74	0.808	0.287	313
(b) 40k iNGP iter.	27.72	0.811	0.284	333
(c) Train iNGP during 3DGS	27.73	0.809	0.285	310
(d) Color from GT image	27.73	0.807	0.287	320
(e) Prediction of SH with iNGP	27.74	0.805	0.290	320
(f) Cone-sized initialization	27.38	0.806	0.287	437
(g) Without initialization	27.46	0.811	0.285	415
(h) Uniform image-space sampling	27.51	0.806	0.286	307
(i) Densify with 3DGS depth	27.43	0.797	0.296	332
(j) SfM initialization + 3DGS depth dens.	27.15	0.790	0.302	329
(k) Densify with k-NN scaling	27.54	0.802	0.295	299
(l) No opacity penalty	27.31	0.794	0.301	294
(m) Post-densification opacity decrease [5]	27.46	0.798	0.297	256
(n) MCMC-style opacity penalty [25]	27.49	0.803	0.293	239
(o) $\lambda_{\text{scale}} = 1$	27.70	0.810	0.285	329
(p) $\lambda_{\text{scale}} = 4$	27.63	0.808	0.285	329

Table 3. **Ablation study** of our method with 100k Gaussians, averaged over the Mip-NeRF360 [3] dataset. We highlight the best, second best, and third best results.

3DGS depth **(i)**, also without using iNGP even for initialization **(j)**, strongly affects the results. Similarly, k-NN sizing of newly added primitives based on their closest neighbors **(k)**, or changing the opacity penalty **(l), (m), (n)**, has a large effect on reconstruction quality, reinforcing the benefits of our densification approach. **(o), (p)** Altering Gaussian sizes from their default pixel-width cone size during densification slightly reduces quality, yet the small difference suggests Gaussians quickly resize to fit the scene.

6. Conclusion

We introduce **ConeGS**, a reconstruction pipeline replacing cloning-based densification with a method guided by photometric error and a coarse iNGP proxy, where new primitives are sized by pixel cones. Together with an improved opacity penalty, this allows creating primitives independently of existing structures through more flexible exploration. ConeGS consistently improves reconstruction quality and rendering performance across Gaussian budgets, with strong gains under tight primitive constraints. It achieves up to 0.6 PSNR increase and 20% speedup over cloning-based baselines.

Limitations: ConeGS performs well on standard scenes but may struggle with large-scale environments, inaccurate poses, or sparse views, occasionally creating floaters (see appendix). Benefits are also reduced at high Gaussian budgets, where dense coverage limits the impact of error-guided placement, primarily offering faster rendering.

Acknowledgments

Stefano Esposito acknowledges travel support from the European Union's Horizon 2020 research and innovation program under ELISE Grant Agreement No. 951847.

References

[1] Mohamed Abdul Gafoor, Marius Preda, and Titus Zaharia. Refining gaussian splatting: A volumetric densification approach. In *Computer Science Research Notes*. University of West Bohemia, Czech Republic, 2025. 3

[2] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *ICCV*, 2021. 2, 3

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 1, 2, 6, 7, 8

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *ICCV*, 2023. 2

[5] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. Revising densification in gaussian splatting. *ArXiv*, abs/2404.06109, 2024. 3, 6, 8

[6] David Charatan, Sizhe Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*, 2024. 3

[7] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022. 2

[8] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. *arXiv preprint arXiv:2403.14627*, 2024. 3

[9] Kai Cheng, Xiaoxiao Long, Kaizhi Yang, Yao Yao, Wei Yin, Yuexin Ma, Wenping Wang, and Xuejin Chen. GaussianPro: 3d gaussian splatting with progressive propagation. In *International Conference on Machine Learning (ICML)*, 2024. 3, 6, 7

[10] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps, 2023. 3

[11] Guangchi Fang and Bing Wang. Mini-splatting2: Building 360 scenes within minutes via aggressive gaussian densification. *ArXiv*, abs/2411.12788, 2024. 3, 6, 7

[12] Guangchi Fang and Bing Wang. Mini-splatting: Representing scenes with a constrained number of gaussians, 2024. 3

[13] Shuangkang Fang, I-Chao Shen, Takeo Igarashi, Yufeng Wang, ZeSheng Wang, Yi Yang, Wenrui Ding, and Shuchang Zhou. Nerf is a valuable assistant for 3d gaussian splatting, 2025. 3

[14] Yalda Foroutan, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. Evaluating alternatives to sfm point cloud initialization for gaussian splatting. 2024. 3, 6, 7

[15] Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. Eagles: Efficient accelerated 3d gaussians with lightweight encodings, 2024. 3

[16] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *CVPR*, 2024. 3

[17] Zhihao Guo, Jingxuan Su, Shenglin Wang, Jinlong Fan, Jing Zhang, Li Hong Han, and Peng Wang. Gp-gs: Gaussian processes for enhanced gaussian splatting. *ArXiv*, abs/2502.02283, 2025. 3

[18] Abdullah Hamdi, Luke Melas-Kyriazi, Jinjie Mai, Guocheng Qian, Ruoshi Liu, Carl Vondrick, Bernard Ghanem, and Andrea Vedaldi. Ges : Generalized exponential splatting for efficient radiance field rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19812–19822, 2024. 3

[19] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. 2018. 6, 7

[20] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 2, 3

[21] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. 2024. 3

[22] Binxiao Huang, Zhengwu Liu, and Ngai Wong. Decomposing densification in gaussian splatting for faster 3d scene reconstruction, 2025. 3

[23] Hanqing Jiang, Xiaojun Xiang, Han Sun, Hongjie Li, Liyang Zhou, Xiaoyu Zhang, and Guofeng Zhang. Geotexdensifier: Geometry-texture-aware densification for high-quality photorealistic 3d gaussian splatting, 2024. 3

[24] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 2, 3, 6, 7, 8

[25] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024. 1, 3, 5, 6, 7, 8

[26] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 1, 6, 7

[27] Dmytro Kotovenko, Olga Grebenkova, and Björn Ommer. EDGS: eliminating densification for efficient convergence of 3dgs. *arXiv*, 2504.13204, 2025. 1, 3, 6, 7, 8

[28] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. *arXiv preprint arXiv:2403.06912*, 2024. 3

[29] Mingrui Li, Shuhong Liu, Tianchen Deng, and Hongyu Wang. Densesplat: Densifying gaussian splatting slam with neural radiance prior, 2025. 3

[30] Ruilong Li, Hang Gao, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: Efficient sampling accelerates nerfs. *arXiv preprint arXiv:2305.04966*, 2023. 6

[31] Rong Liu, Dylan Sun, Meida Chen, Yue Wang, and Andrew Feng. Deformable beta splatting, 2025. 3

[32] Yueh-Cheng Liu, Lukas Höllein, Matthias Nießner, and Angela Dai. Quicksplat: Fast 3d surface reconstruction via learned gaussian initialization. *ArXiv*, abs/2505.05591, 2025. 3

[33] Chongshan Lu, Fukun Yin, Xin Chen, Tao Chen, Gang YU, and Jiayuan Fan. A large-scale outdoor multi-modal dataset and benchmark for novel view synthesis and implicit scene reconstruction, 2023. 1, 6, 7, 8

[34] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *CVPR*, 2024. 2, 3

[35] Alexander Mai, Peter Hedman, George Kopanas, Dor Verbin, David Futschik, Qiangeng Xu, Falko Kuester, Jon Barron, and Yinda Zhang. Ever: Exact volumetric ellipsoid rendering for real-time view synthesis, 2024. 3

[36] Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, New York, NY, USA, 2024. Association for Computing Machinery. 3, 6

[37] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2, 3

[38] Mahmud A. Mohamad, Gamal Elghazaly, Arthur Hubert, and Raphael Frank. Denser: 3d gaussians splatting for scene reconstruction of dynamic urban environments, 2024. 3

[39] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 2022. 2, 4

[40] Seungtae Nam, Xiangyu Sun, Gyeongjin Kang, Younggeun Lee, Seungjun Oh, and Eunbyung Park. Generative densification: Learning to densify gaussians for high-fidelity generalizable 3d reconstruction. *arXiv preprint arXiv:2412.06234*, 2024. 3

[41] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. In *CVPR*, 2024. 3

[42] Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico Tombari. Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps. *arXiv.org*, 2024. 3, 4

[43] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering. *ACM Transactions on Graphics*, 4 (43), 2024. 3

[44] Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2024. 3

[45] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 2

[46] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 2

[47] Chinmay Talegaonkar, Yash Belhe, Ravi Ramamoorthi, and Nicholas Antipa. Volumetrically consistent 3d gaussian rasterization, 2025. 3

[48] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023. 2

[49] Nicolas von Lützow and Matthias Nießner. Linprim: Linear primitives for differentiable volumetric rendering. *ArXiv*, abs/2501.16312, 2025. 2, 3

[50] Zipeng Wang and Dan Xu. Pygs: Large-scale scene representation with pyramidal 3d gaussian splatting, 2024. 3

[51] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004. 3, 6

[52] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *CVPR*, 2024. 2

[53] Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. Multi-scale 3d gaussian splatting for anti-aliased rendering. In *CVPR*, 2024. 2

[54] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023. 2

[55] Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. AbsGS: Recovering fine details in 3d gaussian splatting. In *ACM MM*, 2024. 3

[56] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 2

[57] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. *CVPR*, 2024. 2, 3

[58] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient high-quality compact surface reconstruction in unbounded scenes. *arXiv:2404.10772*, 2024. 3

[59] Zhaojie Zeng, Yuesong Wang, Lili Ju, and Tao Guan. Frequency-aware density control via reparameterization for high-quality rendering of 3d gaussian splatting. *ArXiv*, abs/2503.07000, 2025. 3

[60] Jiahui Zhang, Fangneng Zhan, Muyu Xu, Shijian Lu, and Eric P. Xing. Fregs: 3d gaussian splatting with progressive frequency regularization. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21424–21433, 2024. 3

[61] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 6

[62] Xin Zhang, Anpei Chen, Jincheng Xiong, Pinxuan Dai, Yujun Shen, and Weiwei Xu. Neural shell texture splatting: More details and fewer primitives. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. 3

[63] Yangming Zhang, Wenqi Jia, Wei Niu, and Miao Yin. Gaussianspa: An "optimizing-sparsifying" simplification framework for compact and high-quality 3d gaussian splatting. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26673–26682, 2024. 3

[64] Hongbi Zhou and Zhangkai Ni. Perceptual-gs: Scene-adaptive perceptual densification for gaussian splatting. *ArXiv*, abs/2506.12400, 2025. 3, 6, 7

[65] Zheng Zhou, Yu-Jie Xiong, Chun-Ming Xia, Jia-Chen Zhang, and Hong-Jian Zhan. Gradient-direction-aware density control for 3d gaussian splatting. 2025. 3

[66] Yuanchen Guo Yangguang Li Ding Liang Yanpei Cao Songhai Zhang Zixin Zou, Zhipeng Yu. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. *arXiv preprint arXiv:2312.09147*, 2023. 2

[67] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS '01.*, 2001. 3