3D Gaussian Splatting for Arbitrary Resolutions with Compact Proxy Anchors

Supplementary Materials

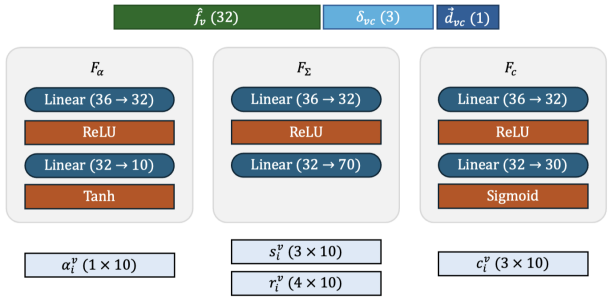

Figure 7. Overview of the Gaussian property prediction MLPs. Three distinct MLPs share the same 36-dimensional input but each outputs a different Gaussian propeties.

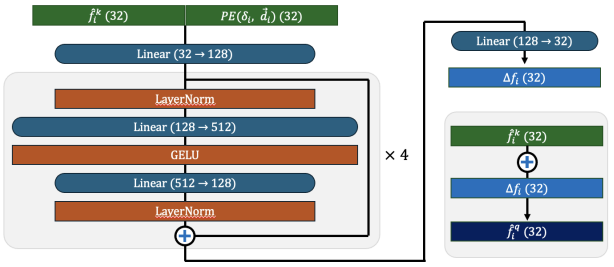

Figure 8. Structure for Decoder. Proxy anchor encoder architecture generating residual anchor features and the stack of residual MLP blocks for prediction.

6. Overview

The Supplementary Materials comprise three parts: Architectural Details, Additional Analysis, and Multi-Resolution Performance. Architectural Details outlines the model structure, layer settings, and additional equations. Additional Analysis extends the ablation study with further experiments assessing each module's role. Together, these sections deepen understanding of our method. Multi-Resolution Performance reports quantitative metrics and qualitative examples at different resolutions.

7. Architectural Details

7.1. Model Structure

Figure 7 illustrates how, for each anchor v, the 32-dimensional feature $\hat{f}_v$, 3-dimensional distance difference δ_{vc}, and 1-dimensional direction vector $\vec{d}_{vc}$ are concatenated into a 36-dimensional input. Three separate MLPs then process this input to predict different properties:

- $\boldsymbol{F_\alpha}$ takes a 36-dimensional input vector and outputs 10-dimensional attention weights $\alpha_i^v \in \mathbb{R}^{10}$.
- $\boldsymbol{F_\Sigma}$ takes the same 36-dimensional input and outputs a 70-dimensional message summary (interpreted as 3×10 scalar and 4×10 vector summaries).
- $\boldsymbol{F_c}$ also takes the 36-dimensional input and outputs a 30-dimensional context update vector $c_i^v \in \mathbb{R}^{30}$.

Figure 8 illustrates the module that predicts a leaf anchor's feature from a proxy anchor. For each leaf anchor, the input consists of the feature of its nearest proxy anchor, $\hat{f}_i^k \in \mathbb{R}^{32}$, and a positional encoding of the relative direction and distance between the proxy and the leaf, $PE(\Delta x_i, \vec{d}_i)$. These inputs are concatenated and passed through an encoder to produce a residual feature, Δf_i. The final prediction is computed in a residual manner as $\hat{f}_i^q$. In other words, the model treats the proxy feature as a baseline and learns only the necessary correction, which encourages a compact proxy anchor representation.

7.2. Random Resolution Sampling During Training

Figure 9 shows record of resolutions used for training iterations. During training, we randomly select a resolution ratio $\xi \in [0.1, 1.0]$ at every iteration and resize the input accordingly. The left plot is the time-ordered record, showing that diverse resolutions are continuously exposed over the course of training. The right plot sorts the same values in ascending order, yielding an almost linear trend that suggests uniform sampling. This design enables the model to learn across the full resolution range without bias.

7.3. Density-Aware Proxy Anchor Selection

In our method, we utilize a set of anchor $V = \{ v \in \mathbb{R}^3 \}$ and partition them into a grid with cell size ε. Each anchor v_i is assigned to a grid cell by computing its index g_i. All anchors whose rounded index equals l form the cell group:

$$g_i = \left\lfloor \frac{v_i}{\varepsilon} \right\rceil, \quad G_l = \{ v_i \mid g_i = l \}. \tag{15}$$

To obtain a density-aware subset of proxy anchors, we sample $\lceil \rho \, |G_l| \rceil$ points uniformly at random from each G_l, where $\rho = 0.3$ denotes a 30% sampling rate. Finally, by taking the union of the sampled anchors across all occupied cells, we define

$$K = \bigcup_l \text{Sample}\big(G_l, \lceil \rho \, |G_l| \rceil\big), \tag{16}$$

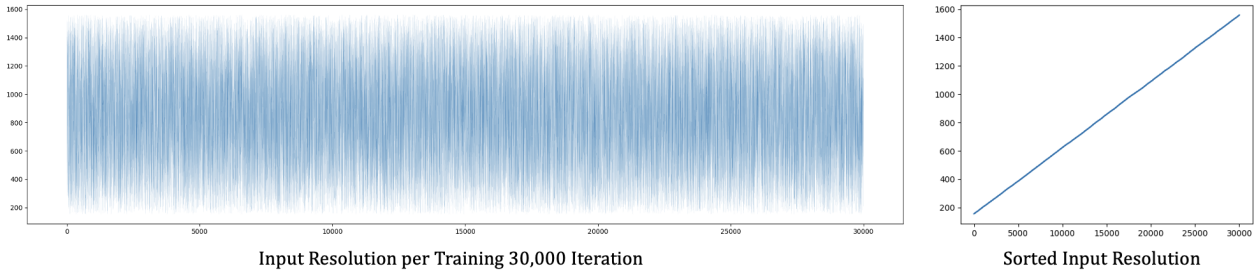

Figure 9. Record of input size changes during training by randomly sampling a resolution ratio $\xi \in [0.1, 1.0]$, when the base image width is 1,559. The left plot shows the widths in training orders, and the right plot shows the same width sorted in ascending order.

Figure 10. Visualization of K-means clustering (K=3) on feature vectors extracted from a fixed set of random 3D points across different trained scene models. The top row shows each scene's ground-truth RGB image, and the bottom row shows the same sampled points projected onto the image plane with cluster assignments indicated by different colors.

which yields the proxy anchor set K retaining a representative 30% of anchors. This ensures that denser regions contribute proportionally more proxy anchors, while maintaining a uniform overall sparsity.

7.4. Implementation Details

In our arbitrary-resolution training protocol (10%–100%), the *100% reference resolution* is defined after applying the standard image rescaling rule used in the original 3D Gaussian Splatting (3D-GS) pipeline [11]. Concretely, under the default preprocessing, if the input image width exceeds 1600 pixels, the image is downscaled so that its width becomes at most 1600. Otherwise, it is kept unchanged. Following this design principle prevents the reference resolution from becoming excessively large while also ensuring that the 10% endpoint falls into a sufficiently ultra-low-resolution regime, which matches our goal of learning under arbitrary resolution conditions. All random resizings between 10% and 100% during training are then applied *relative to this reference resolution* in other words, the post-rescaling "100%" input. For NeRF-Synthetic (8 scenes), we use a rescaling factor of 1, so the original 800×800 images directly serve as the 100% reference resolution. For Tanks&Temples (2 scenes) and DeepBlending (2 scenes), the original image widths do not exceed 1600, hence we also fix the rescaling factor to 1. For Mip-NeRF360, we apply scene-dependent rescaling, indoor scenes (*counter, bonsai, kitchen, room*) with widths around $\sim$3000 use a rescaling factor of 2, whereas outdoor scenes (*flowers, bicycle, garden, treehill, stump*) with widths around $\sim$5000 use a rescaling factor of 4.

7.5. Gradients of the Pixel Coverage Gate

Let L be the loss. For a projected 2×2 covariance $\Sigma_{2D}^{\xi v}$, (7) defines the coverage $A^{\xi v}$, (8) defines the pixel coverage gate $g_{\text{pix}}(A^{\xi v})$, and (9) defines the gated opacity $\tilde{\alpha}_i^{\xi v}$. For readability, we omit superscripts and write $\Sigma = \Sigma_{2D}^{\xi v}$, $A = A^{\xi v}$, $g = g_{\text{pix}}(A^{\xi v})$, and $\tilde{\alpha} = \tilde{\alpha}_i^{\xi v}$, and we assume a sigmoid gate $g(A) = \mathit{sigmoid}(\kappa(A - \eta))$ with slope κ and coverage threshold η. Since $\tilde{\alpha} = \alpha\, g$ and α and g are treated as independent, the opacity gradient is

$$\frac{\partial L}{\partial \alpha} = \frac{\partial L}{\partial \tilde{\alpha}} \frac{\partial \tilde{\alpha}}{\partial \alpha} = \frac{\partial L}{\partial \tilde{\alpha}}\, g. \tag{17}$$

By the chain rule, the gradient with respect to the coverage A becomes

$$\frac{\partial L}{\partial A} = \frac{\partial L}{\partial \tilde{\alpha}} \frac{\partial \tilde{\alpha}}{\partial g} \frac{\mathrm{d}g}{\mathrm{d}A} = \frac{\partial L}{\partial \tilde{\alpha}}\, \alpha\, \kappa\, g(1-g). \tag{18}$$

Moreover, using $A = \pi\sqrt{\det(\Sigma)}$ and $\partial \det(\Sigma)/\partial \Sigma = \det(\Sigma)\, \Sigma^{-T}$, we obtain

$$\frac{\partial A}{\partial \Sigma} = \frac{\pi}{2} \sqrt{\det(\Sigma)}\, \Sigma^{-T}. \tag{19}$$

Therefore, combining the above via the chain rule yields the covariance gradient

$$\frac{\partial L}{\partial \Sigma} = \frac{\partial L}{\partial A} \frac{\partial A}{\partial \Sigma} = \frac{\partial L}{\partial \tilde{\alpha}}\, \alpha\, \kappa\, g(1-g)\, \frac{\pi}{2} \sqrt{\det(\Sigma)}\, \Sigma^{-T}. \tag{20}$$

The gradients derived above are additive contributions that accumulate with the existing $\frac{\partial L}{\partial \alpha}$ and $\frac{\partial L}{\partial \Sigma}$ terms and are propagated jointly by standard backpropagation.

Tanks&Temples												
Resolution	10%		30%		50%		70%		90%		Average	
Metric	PSNR(↑)	LPIPS(↓)	PSNR(↑)	LPIPS(↓)	PSNR(↑)	LPIPS(↓)	PSNR(↑)	LPIPS(↓)	PSNR(↑)	LPIPS(↓)	PSNR(↑)	LPIPS(↓)
Proxy&Leaf (MLP)	23.824	0.072	24.219	0.114	23.330	0.182	22.777	0.241	22.458	0.282	23.276	0.191
Proxy&Leaf (Cross-Att)	23.143	0.084	23.908	0.133	23.143	0.190	22.469	0.259	21.971	0.300	22.692	0.211
Proxy&Leaf (Self-Att)	23.024	0.193	23.846	0.140	23.327	0.185	22.392	0.253	22.002	0.298	22.681	0.209
Residual+FiLM	24.751	0.060	25.551	0.079	24.613	0.134	23.970	0.186	23.603	0.227	24.465	0.148
Residual (**Ours**)	**24.889**	**0.059**	**25.689**	**0.076**	**24.757**	**0.131**	**24.109**	**0.182**	**23.742**	**0.222**	**24.600**	**0.144**

DeepBlending												
Resolution	10%		30%		50%		70%		90%		Average	
Metric	PSNR(↑)	LPIPS(↓)	PSNR(↑)	LPIPS(↓)	PSNR(↑)	LPIPS(↓)	PSNR(↑)	LPIPS(↓)	PSNR(↑)	LPIPS(↓)	PSNR(↑)	LPIPS(↓)
Proxy&Leaf (MLP)	30.321	0.057	30.028	0.137	29.488	0.214	29.261	0.271	29.157	0.296	29.606	0.210
Proxy&Leaf (Cross-Att)	29.482	0.061	29.104	0.152	28.526	0.209	27.915	0.288	27.678	0.308	28.541	0.224
Proxy&Leaf (Self-Att)	29.334	0.060	29.390	0.150	28.512	0.210	27.792	0.290	27.512	0.305	28.508	0.228
Residual+FiLM	**30.483**	0.053	30.336	0.122	29.787	0.199	29.548	0.256	29.435	0.284	29.884	0.197
Residual (**Ours**)	30.472	**0.053**	**30.377**	**0.118**	**29.814**	**0.196**	**29.558**	**0.254**	**29.453**	**0.282**	**29.905**	**0.194**

Table 5. Ablation study on the Proxy Anchor Encoder Structure. Proxy&Leaf takes both proxy and leaf jointly as input to predict the leaf output, whereas Residual variants take only the Proxy as input and predict the Proxy→leaf residual (Δ). Residual (Ours) achieves the most stable performance across resolutions and datasets.

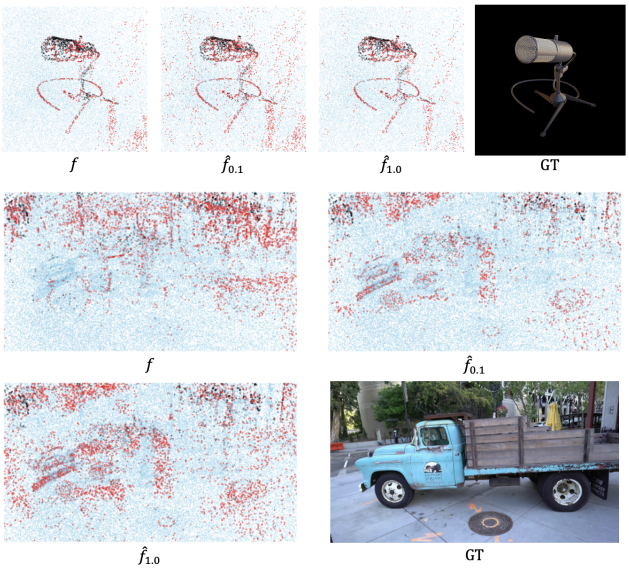

Figure 11. K-means clustering (K=3) of interpolated features with and without resolution-embedding on two benchmark scenes. Top row: NeRF-Synthetic *mic* dataset; bottom row: Tanks&Temples *truck* dataset. From left to right in each row: clusters for the raw interpolated feature f, for the enhanced feature $\hat{f}$ with a resolution-embedding scale of 0.1, and for $\hat{f}$ with a scale of 1.0. The rightmost column shows the corresponding ground-truth renderings (GT).

8. Additional Analysis

8.1. 3D Anchor Representation

In this section, we validate a representational benefit of our framework beyond the compression gain provided by the residual anchor predictor. Conventional Gaussian-based or anchor-based methods store a scene as an explicit set of points after training, making it difficult to assign meaningful scene information to 3D locations that are not explicitly stored. In contrast, our approach retains only proxy anchors and uses the residual anchor predictor to infer surrounding leaf anchors, thereby enabling plausible feature synthesis at arbitrary 3D locations under the assumption that an anchor can be formed there. To verify this, we extract the predicted anchor features at an identical set of random 3D query points after training and compare their distributions via K-means clustering (K=3). Moreover, by injecting FiLM-based resolution embeddings and repeating the same analysis at different resolutions, we observe resolution-adaptive shifts in the feature space, indicating that the resolution signal induces changes in the learned representation.

Anchor Feature for Random Points. Figure 10 presents the result of K-means clustering on anchor features located at random 3D points. Figure 10 shows K-means clustering (K=3) of predicted anchor features at random 3D locations. Each trained scene model yields distinct semantic clusters from identical spatial inputs, indicating scene-specific feature organization. Each model produced unique semantic clusters despite identical spatial inputs. This contrasts with traditional Gaussian Splatting, which only learns explicit point attributes, showing that our implicit position encoding enables dynamic, scene-specific feature representation.

Anchor Feature Variation Across Resolutions. Figure 11 presents an extension of the experiments introduced in 3D Anchor Representation of ablation study. First, we

	Dataset (*Metric: PSNR*)			
$\mathcal{L}_{recon}$	Mip360	DB	NeRF-Syn	T&T
MSE	28.009	29.606	**31.184**	24.480
L1	27.391	29.541	29.931	23.817
Huber (L1+L2)	27.464	29.623	29.997	22.712
MSE+cos **(Ours)**	**28.183**	**29.905**	31.056	**24.600**

	Dataset (*Metric: LPIPS*)			
$\mathcal{L}_{recon}$	Mip360	DB	NeRF-Syn	T&T
MSE	0.197	0.205	**0.056**	0.162
L1	0.220	0.236	0.101	0.217
Huber (L1+L2)	0.216	0.229	0.088	0.201
MSE+cos **(Ours)**	**0.191**	**0.194**	0.058	**0.144**

Table 6. Ablation study on the Mip-NeRF360, DeepBlending, NeRF-Synthetic and Tanks&Temples dataset. Resolution Average results are reported under the same experimental settings as in Table 2, comparing performance with reconstruction loss objective.

apply K-means clustering (K=3) to the raw interpolated feature f sampled at random 3D points, replicating the basic 3D Anchor Representation study. We inject FiLM-based resolution embeddings into the Gaussian-properties MLP and cluster features at two scales (0.1, 1.0). Figure 11 visualizes consistent, resolution-driven shifts in the feature distribution. This confirms that multi resolution-embedding enriches the feature representation, enabling the model to capture finer geometric details.

8.2. Ablation on Losses.

In Table 6, we compare reconstruction objectives by reporting resolution averaged PSNR and LPIPS on whole datasets (Mip-NeRF360, DeepBlending, NeRF-Synthetic and Tanks&Temples). We ablate the reconstruction loss in (13) by training four variants that compute the discrepancy between the predicted leaf features $\tilde{F}_q^{\in}$ and targets $\hat{F}_q^{\in}$ using a single objective: (i) MSE, (ii) L1, (iii) Huber (L1+L2), and (iv) our combined MSE+cosine loss. The results show that the proposed objective consistently achieves the best or competitive performance across datasets, in terms of PSNR and LPIPS on Mip-NeRF360, DeepBlending, and Tanks&Temples, while remaining close to the top result on NeRF-Synthetic. This is likely because NeRF-Synthetic contains simple, background-free scenes, where a pure MSE objective performs particularly well.

8.3. Predictor Module Ablation

Table 5 compares predictor designs for generating leaf anchors from proxy anchors. First, the Proxy&Leaf mirrors the MAE setup: the target leaf (masked) and the reference Proxy (unmasked) are fed jointly to reconstruct the leaf output. Within this scheme, we evaluate (i) a simple MLP block, (ii) Cross-Attention (mutual attention between leaf and proxy), and (iii) Self-Attention (self-attention over the combined set of proxy and leaf tokens). In contrast,

Datasets	Mip-NeRF360								
Resolutions	20%		60%		100%		Avg		
Method	PSNR(↑)	LPIPS(↓)	PSNR(↑)	LPIPS(↓)	PSNR(↑)	LPIPS(↓)	PSNR(↑)	LPIPS(↓)	#GS(↓)
3D-GS	24.705	0.158	24.884	0.252	26.565	0.311	25.025	0.226	1.29M
Mip-splat	27.431	0.121	26.519	0.227	26.822	0.272	26.868	0.209	1.10M
Analytic-splat	27.467	0.129	26.317	0.232	26.801	0.279	26.851	0.208	1.05M
Scaffold-GS	24.636	0.161	24.825	0.262	26.518	0.316	25.192	0.230	1.06M
Ours-No $\mathcal{E}_\xi$	25.663	0.135	25.381	0.249	26.382	0.287	25.447	0.222	1.10M
Ours-$\mathcal{E}_\xi$	23.796	0.150	23.026	0.271	26.881	0.276	24.123	0.231	1.14M
Ours-FiLM$_\xi$	28.037	0.105	26.913	0.217	26.819	0.277	27.115	0.194	1.08M

Table 7. Single-train/multi-test setting on Mip-NeRF360. We report PSNR/LPIPS at 20%, 60%, 100%, average and #GS.

Mip-NeRF360 (*Metric: PSNR*)								
Resolution	2.5%	5%	7.5%	110%	120%	130%	140%	150%
3D-GS	17.060	19.275	20.950	24.359	24.245	24.152	24.084	24.012
Ours-$\mathcal{E}_\xi$	17.320	20.301	27.193	26.477	26.097	25.715	25.325	24.938
Ours-FiLM$_\xi$	**18.435**	**23.528**	**27.577**	**26.539**	**26.301**	**26.009**	**27.132**	**25.690**

Table 8. Quantitative results at out of range resolutions (2.5–7.5% and 110–150%) on Mip-NeRF360 under matched Gaussian counts.

the Residual does not feed proxy and leaf together, it takes only the proxy as input and directly predicts the proxy→leaf residual. We also validate a FiLM-based, resolution-aware variant (Residual+FiLM). Overall, attention-based designs incur substantial computational overhead yet yield limited improvements over a simple MLP. By contrast, Ours improves performance consistently with a lightweight architecture, achieving the highest average results across all resolution levels and the two benchmarks (Tanks&Temples and DeepBlending). This indicates that learning the residual from proxy to leaf provides a sufficiently strong supervisory signal while striking a favorable balance between accuracy and efficiency.

8.4. Resolution Robustness Comparison

Table 7 and Table 8 evaluate how robust each method is to test-time resolution changes. In particular, we highlight that FiLM-based resolution embedding substantially improves robustness under both single-train/multi-test and (ii) resolution extrapolation settings. Table 7 reports PSNR/LPIPS on Mip-NeRF360 when training is conducted at a single resolution while evaluation spans multiple resolutions (20%, 60%, 100%), under matched Gaussian counts. Our method maintains consistently stronger reconstruction quality across all tested resolutions, and the FiLM-based resolution embedding variant achieves the best overall trade-off. These results indicate that explicitly conditioning the renderer on resolution enables stable performance even when the test resolution differs from the training resolution. To further test resolution extrapolation, Table 8 evaluates models trained on the proposed multi-resolution protocol (10%–100%) at unseen lower resolutions (2.5%, 5%, 7.5%) and super-resolution regimes (110%–150%), again under matched Gaussian counts. Despite these test resolutions lying outside the training range, our method remains

Training Time (sec, ↓)			
Dataset	Mip360	T&T	DB
Context-GS	5297.9	2469.9	2141.6
TC-GS	3207.2	2588.4	1980.3
Ours	2163.7	1759.8	1473.1

Table 9. Training-time comparison between two anchor-based compression baselines (Context-GS and TC-GS) and our method, measured on Mip-NeRF360, Tanks&Temples ("T&T"), and Deep-Blending ("DB") with a fixed number of Gaussians.

	T&T		NeRF-Syn	
λ_{recon}	PSNR (↑)	SSIM (↑)	PSNR (↑)	SSIM (↑)
$\text{fixed}_{0.1}$	24.451	0.899	31.007	0.948
$\text{fixed}_{0.01}$	24.194	0.889	**31.062**	0.952
Ours	**24.600**	**0.906**	31.056	**0.954**

Table 10. Ablation of the reconstruction-loss weight λ_{recon} on Tanks&Temples (T&T) and NeRF-Synthetic (NeRF-Syn). We compare two fixed settings (λ_{recon}=0.1 and 0.01) against ours, which uses λ_{recon}=0.1 until 20k iterations and switches to 0.01 thereafter.

Mip-NeRF360				
	$\kappa = 4$		$\kappa = 5$	
η	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
0.80	26.724	0.229	26.731	0.229
0.50	27.544	0.218	27.481	0.220
0.33	28.007	0.201	28.102	0.200
0.25	**28.183**	**0.191**	28.175	0.195
0.10	27.849	0.209	27.950	0.209

NeRF-Synthetic				
	$\kappa = 4$		$\kappa = 5$	
η	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
0.80	28.519	0.104	28.472	0.113
0.50	29.271	0.092	29.181	0.102
0.33	30.497	0.076	30.518	0.085
0.25	**31.056**	**0.049**	31.002	0.051
0.10	31.020	0.060	30.867	0.063

Table 11. Ablation on the hyperparameters κ and η used in the definition of the pixel coverage gate, evaluated on Mip-NeRF360 and NeRF-Synthetic with PSNR and LPIPS as metrics.

consistently superior to 3D-GS, and the FiLM-based resolution embedding achieves the highest PSNR across all unseen resolutions. This demonstrates that our resolution-aware design is not merely interpolating within the observed scale range, but provides **robust generalization beyond the training band**, validating FiLM-based resolution embedding as a key component for resolution robustness.

8.5. Training Time Comparison

Table 9 reports end-to-end training times for the two anchor based compression baselines (Context-GS and TC-GS) and our method, again using the same number of Gaussian. Despite the added modules, our approach trains as quickly or faster than the baselines on all three datasets, indicating that the performance improvements from our 3D Anchor Representation incur minimal computational overhead.

8.6. Hyper Parameter Ablation

Table 10 compares two fixed settings of λ_{recon} defined in (13), namely 0.1 and 0.01, against our schedule, which uses 0.1 until $20k$ iterations and switches to 0.01 afterward on Tanks&Temples and NeRF-Synthetic. Ours consistently provides a stronger quality than fixed settings, so we adopt it as the default.

Table 11 reports the ablation results for the hyperparameters κ and η defined in (8) of the pixel coverage gate. Here, κ controls the sharpness of the gate, and η denotes the coverage threshold. As stated in the definition, ρ, which denotes the Mahalanobis radius, is fixed to 1, and η is the term that directly controls the coverage, therefore the ablation varies only κ and η. Across the tested combinations, the setting $\kappa = 4$ and $\eta = 0.25$ achieves consistent strong performance on both datasets and is adopted as the default. From a conceptual viewpoint, the pixel coverage gate is a coverage-based sigmoid, so η exerts influence than κ. The results show that variations of η exhibit a clear and monotonic pattern around the selected value, while variations of κ do not present a similar trend in performance.

Figure 12 reports an ablation on Mip-NeRF360 that varies the sampling rate per iteration, that is, the fraction of leaf anchors queried by the Residual Anchor Predictor at each step, to examine the trade-off between performance and training cost. As the sampling rate increases, the predictor is exposed to more diverse anchor structures, yielding a gradual increase in PSNR but a concomitant rise in training time. Observing the early knee of the curve, we find that the 20% setting secures strong performance with modest additional cost, and thus we adopt it as our default.

Figure 13 reports an ablation on Mip-NeRF360 that varies the ratio of proxy anchors to all anchors, analyzing the trade-off between performance and post training storage. As the proxy anchor ratio increases, the share of leaf anchors to be predicted decreases, yielding a gradual improvement in PSNR, while storage increases approximately linearly. Considering the performance elbow and the balance with storage efficiency, we adopt a 30% proxy anchor ratio as the default.

9. Multi-Resolution Performance

Table 12 and Table 13 compare the performance of our method against baselines (3D-GS, Mip-splat, Analytic-splat, Scaffold-GS, Context-GS, and TC-GS) on multi-resolution performance. For each dataset, we evaluate PSNR, SSIM, and LPIPS at ten different levels of random input resolution (from 10% to 100%) and report the average over all settings. These results demonstrate how each approach degrades gracefully (or not) as input resolution

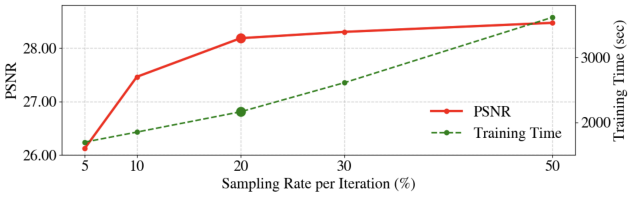

Figure 12. Ablation of the sampling rate per iteration (i.e., the percentage of leaf anchors queried by the residual anchor predictor) on Mip-NeRF360. We plot PSNR (↑, left axis) and training time in seconds (↓, right axis). Increasing the sampling rate yields marginal PSNR gains but increases training time; the 20% setting lies near the early knee of the curve and offers a favorable trade-off, so we adopt it by default.

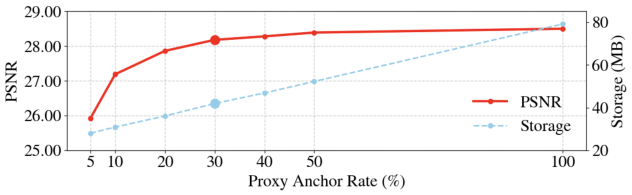

Figure 13. Ablation of the proxy anchor rate on Mip-NeRF360. We plot PSNR (↑, left axis) and storage in MB (↓, right axis). Increasing the sampling rate yields marginal PSNR gains but increases Storage; the 30% setting lies near the early knee of the curve and offers a favorable trade-off, so we adopt it by default.

decreases, highlighting the robustness of our method across both Mip-NeRF360 and DeepBlending benchmarks. These tables provide a detailed per-resolution breakdown of reconstruction quality of Table 1, enabling a finer-grained analysis of each method's behavior under varying input fidelity.

Figure 14 illustrates the two example scenes used for evaluation, with red bounding boxes indicating the local regions extracted for the resolution comparison. Figure 15 and Figure 16 then show qualitative reconstruction error maps for the benchmark methods and our model under varying input resolutions: Figure 15 presents the Train scene from Tanks&Temples, and Figure 16 presents the Stump scene in Mip-NeRF360.

10. Discussion on Future Works

Although our method reconstructs leaf-anchor features from proxy anchors through the residual predictor, it does not completely hide the entire anchor representation. More specifically, while the feature component can be compactly recovered through prediction, other elements such as anchor positions and geometric factors (e.g., scaling factors) still need to be explicitly stored. Therefore, our method should be understood not as a fully replacement-based compression scheme for the entire anchor representation, but rather as an approach that improves storage efficiency primarily

Figure 14. Illustration of the two example scenes used for evaluation. Red bounding boxes indicate the local regions extracted for the qualitative resolution comparison shown in Figure 15 and Figure 16.

along the feature dimension. A promising future direction would be to extend this framework so that not only features, but also positional and other geometric attributes, can be represented in a more compact manner, potentially enabling more fundamental gains in storage efficiency.

In addition, although the comparisons between our method and the benchmark approaches were conducted under controlled settings to validate resolution robustness and compression effectiveness, they do not fully reflect real-world application scenarios. In particular, the current evaluation is performed under settings that maintain a certain level of Gaussian count, and thus may not sufficiently capture the rendering quality and efficiency requirements demanded in practical applications. To extend this work toward real deployment scenarios, it would be important to further examine whether a similar performance efficiency trade-off can be preserved when the Gaussian budget is increased beyond the current evaluation range. From this perspective, the present experimental setting should be viewed as an initial step toward practical application, while evaluation under more realistic conditions and corresponding design improvements remain directions for future work.

Mip-NeRF360

Metrics	PSNR										
Method	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	Avg
3D-GS	21.781	24.971	26.457	27.052	27.212	27.167	27.024	26.841	26.652	26.471	26.163
Mip-splat	28.044	27.798	27.891	27.771	27.529	27.265	27.028	26.830	26.670	26.542	27.337
Analytic-splat	28.449	28.514	28.149	27.708	27.302	26.964	26.695	26.481	26.313	26.180	27.276
Scaffold-GS	24.681	27.693	28.451	28.344	27.979	27.582	27.132	26.935	26.696	26.498	27.199
Context-GS	24.560	27.5838	28.297	28.222	27.823	27.457	27.120	26.740	26.569	26.344	27.072
TC-GS	24.521	27.437	28.196	28.132	27.772	27.311	27.042	26.699	26.528	26.284	26.992
Ours	29.742	30.307	29.446	28.678	28.083	27.637	27.305	27.055	26.865	26.710	28.183

Metrics	SSIM										
Method	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	Avg
3D-GS	0.746	0.813	0.825	0.818	0.803	0.788	0.774	0.760	0.749	0.739	0.782
Mip-splat	0.874	0.845	0.834	0.819	0.804	0.789	0.774	0.763	0.754	0.745	0.800
Analytic-splat	0.895	0.867	0.842	0.819	0.799	0.782	0.768	0.756	0.747	0.739	0.801
Scaffold-GS	0.812	0.864	0.860	0.843	0.823	0.803	0.785	0.770	0.756	0.745	0.806
Context-GS	0.809	0.860	0.855	0.840	0.819	0.799	0.781	0.765	0.752	0.741	0.802
TC-GS	0.808	0.858	0.854	0.839	0.819	0.797	0.780	0.766	0.750	0.740	0.801
Ours	0.932	0.916	0.887	0.857	0.830	0.806	0.787	0.772	0.760	0.749	0.830

Metrics	LPIPS										
Method	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	Avg
3D-GS	0.197	0.160	0.173	0.196	0.222	0.246	0.267	0.286	0.303	0.319	0.237
Mip-splat	0.129	0.156	0.175	0.196	0.219	0.242	0.261	0.279	0.294	0.307	0.226
Analytic-splat	0.107	0.141	0.174	0.203	0.230	0.254	0.274	0.291	0.306	0.319	0.230
Scaffold-GS	0.147	0.125	0.147	0.179	0.213	0.244	0.272	0.296	0.372	0.337	0.228
Context-GS	0.148	0.126	0.148	0.180	0.214	0.246	0.273	0.298	0.320	0.340	0.229
TC-GS	0.148	0.126	0.148	0.180	0.214	0.247	0.273	0.299	0.321	0.339	0.230
Ours	0.062	0.081	0.117	0.154	0.189	0.219	0.244	0.265	0.284	0.300	0.191

Table 12. Results on the Mip-NeRF360 dataset, reporting PSNR, SSIM, and LPIPS over randomly sampled 10%–100% multi-resolution inputs. This table presents the average full-resolution performance of the experiments conducted in Table 1, broken down by individual input resolution; all benchmark models and Ours are evaluated under the same #GS and memory settings as in Table 1.

DeepBlending

Metrics	PSNR										
Method	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	Avg
3D-GS	28.747	28.459	28.627	28.659	27.1724	28.282	28.007	27.995	28.729	27.613	28.184
Mip-splat	28.262	28.431	28.350	28.223	28.127	28.044	27.987	27.949	27.920	27.861	28.116
Analytic-splat	29.124	29.345	29.154	28.965	28.821	28.711	28.636	28.584	28.544	28.473	28.836
Scaffold-GS	27.185	29.775	30.211	30.007	29.905	29.749	29.516	29.507	29.425	29.276	29.465
Context-GS	26.704	29.201	29.703	29.622	29.375	29.164	29.161	29.171	28.932	28.790	28.984
TC-GS	26.785	29.370	29.762	29.665	29.429	29.257	29.238	29.108	29.071	28.851	29.054
Ours	30.472	30.859	30.377	30.033	29.814	29.657	29.558	29.497	29.453	29.332	29.905

Metrics	SSIM										
Method	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	Avg
3D-GS	0.883	0.883	0.894	0.884	0.872	0.887	0.889	0.859	0.884	0.874	0.881
Mip-splat	0.900	0.895	0.894	0.892	0.889	0.887	0.887	0.886	0.887	0.885	0.890
Analytic-splat	0.921	0.915	0.910	0.905	0.901	0.896	0.895	0.894	0.894	0.891	0.902
Scaffold-GS	0.877	0.906	0.903	0.897	0.890	0.884	0.880	0.877	0.875	0.870	0.886
Context-GS	0.862	0.894	0.884	0.877	0.874	0.870	0.862	0.862	0.860	0.850	0.870
TC-GS	0.867	0.891	0.889	0.885	0.881	0.868	0.863	0.862	0.862	0.855	0.872
Ours	0.943	0.934	0.920	0.910	0.902	0.896	0.893	0.890	0.888	0.885	0.906

Metrics	LPIPS										
Method	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	Avg
3D-GS	0.088	0.115	0.144	0.178	0.216	0.240	0.262	0.289	0.280	0.312	0.212
Mip-splat	0.107	0.129	0.155	0.185	0.214	0.242	0.263	0.276	0.285	0.302	0.216
Analytic-splat	0.086	0.108	0.138	0.169	0.200	0.230	0.251	0.266	0.276	0.293	0.202
Scaffold-GS	0.086	0.087	0.117	0.162	0.193	0.230	0.256	0.275	0.289	0.311	0.201
Context-GS	0.088	0.088	0.120	0.164	0.195	0.236	0.263	0.282	0.293	0.317	0.205
TC-GS	0.089	0.089	0.120	0.164	0.198	0.236	0.263	0.281	0.296	0.319	0.205
Ours	0.053	0.079	0.115	0.159	0.196	0.230	0.254	0.271	0.282	0.301	0.194

Table 13. Results on the DeepBlending dataset, reporting PSNR, SSIM, and LPIPS over randomly sampled 10%–100% multi-resolution inputs. This table presents the average full-resolution performance of the experiments conducted in Table 1, broken down by individual input resolution; all benchmark models and Ours are evaluated under the same #GS and memory settings as in Table 1.

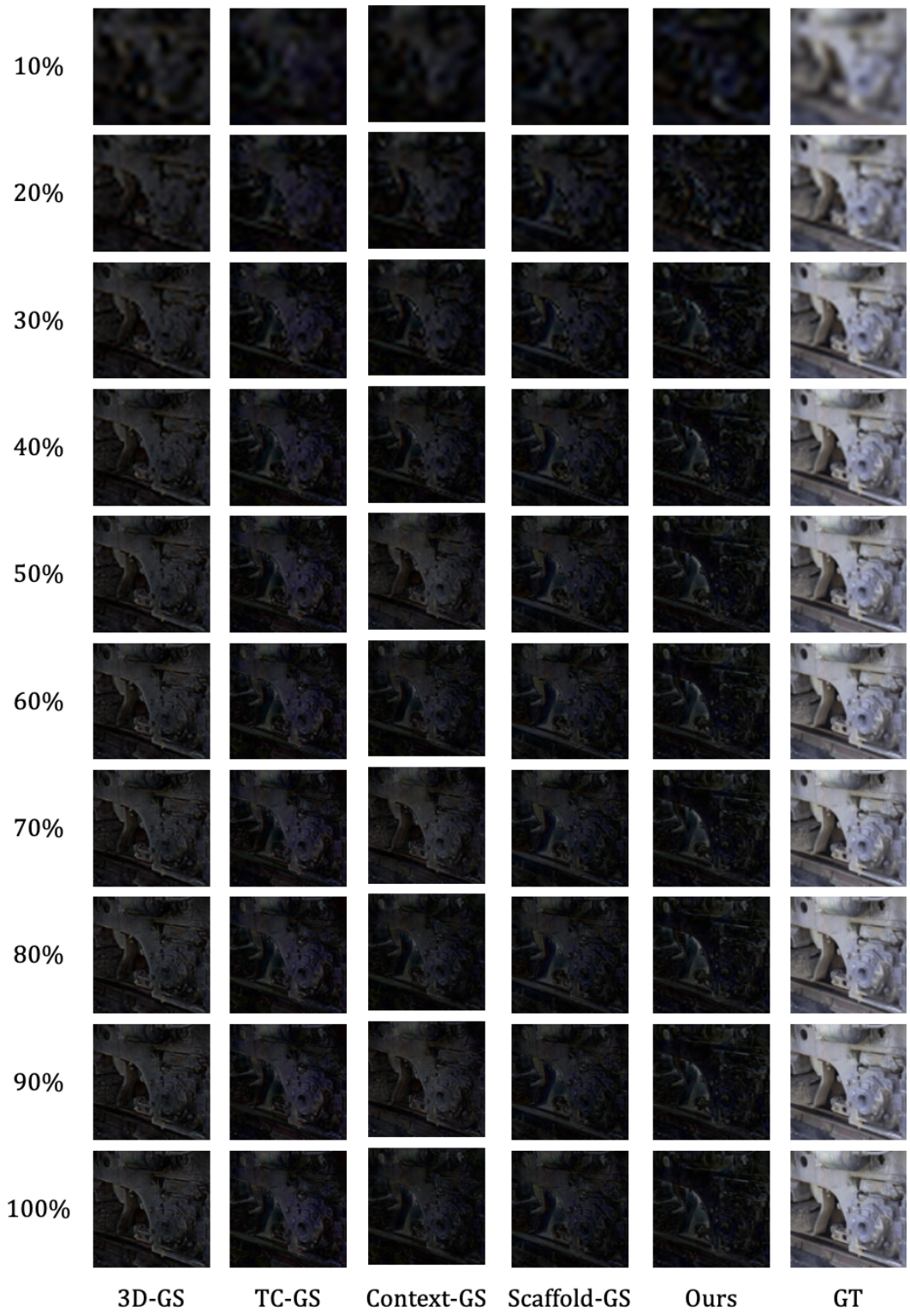

Figure 15. Qualitative comparison of reconstruction error maps on Train scene in Tanks&Temples datasets, for input resolutions ranging from 10% to 100%. Columns 1–4 show error maps for the 3D-GS, TC-GS, Context-GS, and Scaffold-GS benchmarks, column 5 shows the error map of our method, and the rightmost column shows the ground-truth rendering.

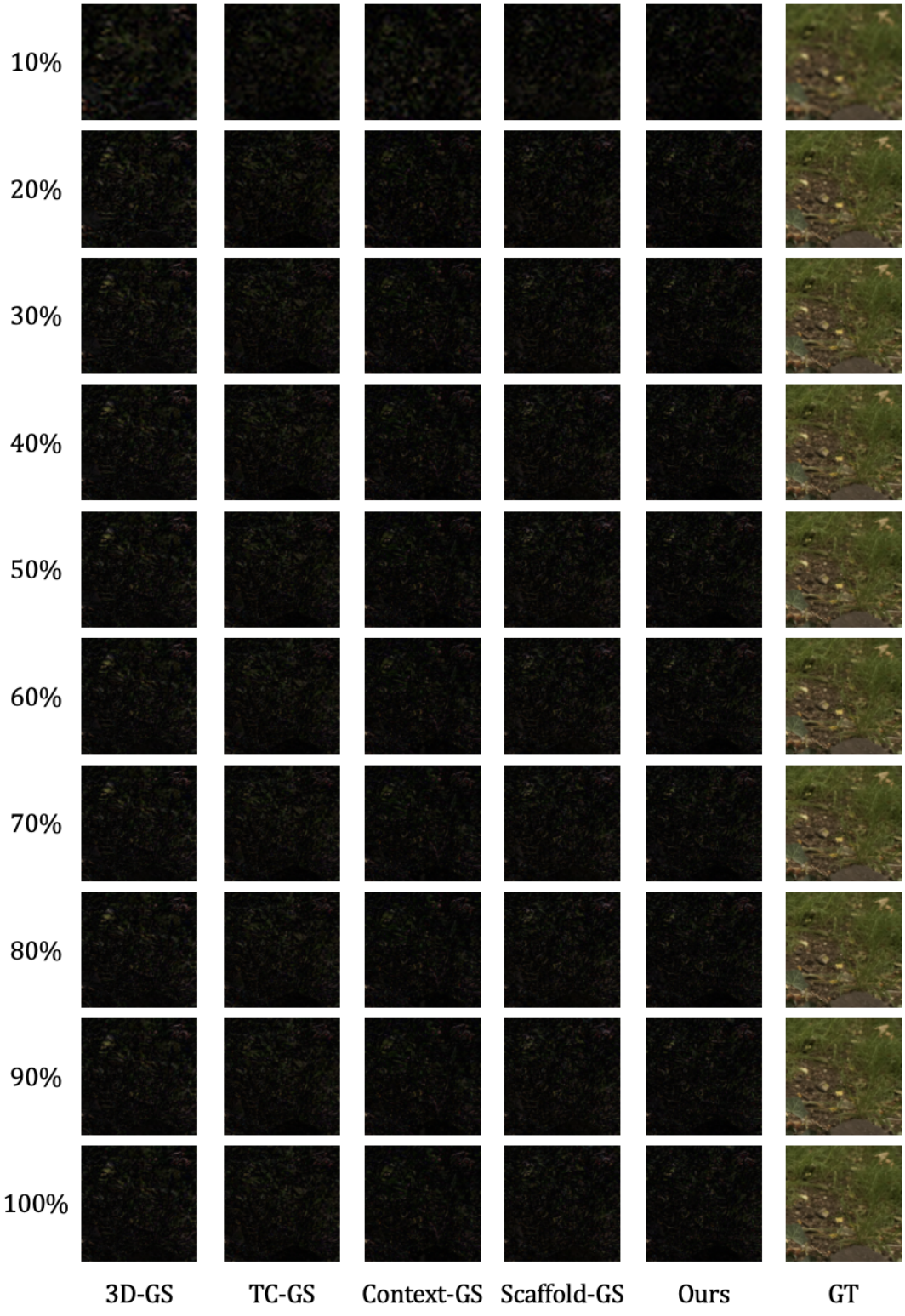

Figure 16. Qualitative comparison of reconstruction error maps on the Stump scene from the Mip-NeRF360 dataset, for input resolutions ranging from 10% to 100%. Columns 1–4 show error maps for the 3D-GS, TC-GS, Context-GS, and Scaffold-GS baselines, column 5 shows the error map of our method, and the rightmost column shows the ground-truth rendering.