LapisGS: Layered Progressive 3D Gaussian Splatting for Adaptive Streaming

Yuang Shi[1,3] Géraldine Morin[2,3] Simone Gasparini[2,3] Wei Tsang Ooi[1,3]

[1]National University of Singapore [2]IRIT - University of Toulouse [3]IPAL, IRL2955, Singapore

{yuangshi, ooiwt}@comp.nus.edu.sg {geraldine.morin, simone.gasparini}@toulouse-inp.fr

Abstract

The rise of Extended Reality (XR) requires efficient streaming of 3D online worlds, challenging current 3DGS representations to adapt to bandwidth-constrained environments. This paper proposes LapisGS, a layered 3DGS that supports adaptive streaming and progressive rendering. Our method constructs a layered structure for cumulative representation, incorporates dynamic opacity optimization to maintain visual fidelity, and utilizes occupancy maps to efficiently manage Gaussian splats. This proposed model offers a progressive representation supporting a continuous rendering quality adapted for bandwidth-aware streaming. Extensive experiments validate the effectiveness of our approach in balancing visual fidelity with the compactness of the model, with up to 50.71% improvement in SSIM, 286.53% improvement in LPIPS with 23% of the original model size, and shows its potential for bandwidth-adapted 3D streaming and rendering applications. Project page: https://yuang-ian.github.io/lapisgs/

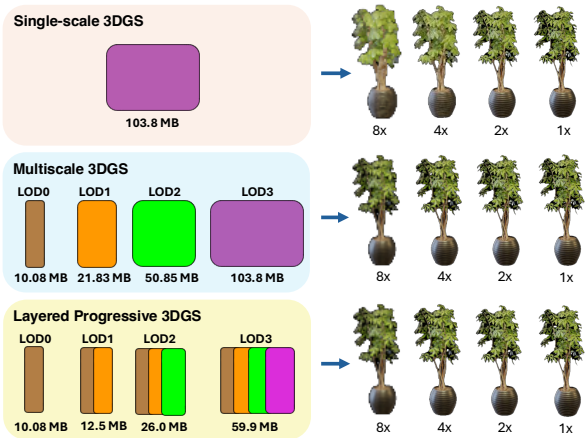


Figure 1. The illustration of the architecture of the single-scale model (upper), multiscale model (middle), and layered progressive model (lower) and their sample renderings at different resolution scales (1×, 2×, 4×, and 8×). Our layered progressive model is tailored for adaptive streaming and view-adaptive rendering.

1. Introduction

In recent years, there has been a significant surge in the popularity of diverse XR applications like virtual reality (VR), augmented reality (AR), and cloud gaming, largely driven by the need to provide users with seamless access to online 3D environments. The success of these applications hinges on the ability to represent complex 3D scenes accurately.

Recently, 3D Gaussian Splatting (3DGS) [11] has emerged as an efficient technique for generating an explicit 3D representation from calibrated images, providing photo-realistic visual quality and supporting fast rendering.

While 3DGS offers advantages for representing and rendering high-fidelity 3D scenes, the resulting data size can be prohibitively large for streaming to users with various bandwidth constraints due to transfer over diverse and dynamic networks and different rendering capacities due to heterogeneous devices. Recent works [5, 7, 14, 17, 19–21, 28] to reduce data size through model compaction or optimization are insufficient to address the dynamic nature of network conditions and device capabilities.

To overcome these limitations, adaptive streaming is crucial to optimize resource utilization, visual quality, and user experience [27]. Specifically, adaptive streaming enables efficient and scalable transmission of 3D content by leveraging a layered representation. This layered structure organizes the 3D data into multiple levels of detail (LOD), typically comprising a base layer and one or more enhancement layers. The base layer provides a basic representation, while the enhancement layers progressively refine the visual quality.

The layered representation must meet specific requirements to effectively support adaptive streaming for 3DGS. Firstly, given subsets of Gaussian splats, the model should be able to represent complete 3D content with reduced details, allowing for the generation of a lower-quality base and enhancement layers. Secondly, progressive levels of detail should inherently share visual information with lower lev-

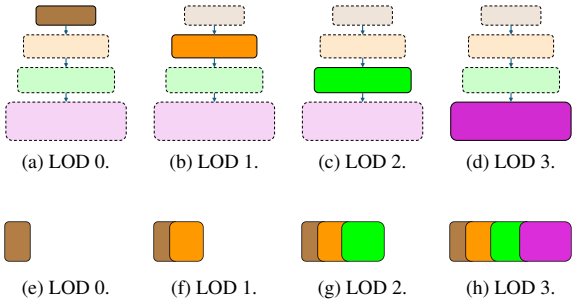


Figure 2. Illustrations of space-based representation characterized by discrete quality layers (upper) and layered progressive representation which comprises incremental layers (lower).

els to minimize redundancy in streaming and support view-adaptive rendering. By leveraging shared visual characteristics across different levels, the progressive representation can reduce the amount of data needed to be transmitted while facilitating smooth and continuous level transition for rendering.

Recent efforts have been made to build hierarchical 3DGS representations based on multiresolution space partitioning to render very large scenes in real time [12, 16, 22]. For instance, Ren *et al*. [22] decompose the 3D space with an octree and optimize Gaussian splats at each octree node, while Kerbl *et al*. [12] regard every splat as a tree node and construct a tree-based hierarchy by recursively merging the neighboring Gaussian splats into parent nodes. However, these space-based hierarchies, while being effective and efficient for local rendering, suffer from the following drawbacks in supporting adaptive streaming:

- Being built based on spatial information, space-based hierarchies do not explicitly define quality levels. Hence, tree traversal algorithms are required to select appropriate splats at different layers to build levels of detail. This added complexity can be computationally expensive and less adaptable to rapidly changing network conditions, potentially leading to inefficient streaming decisions.
- Secondly, each level of detail captured from the space-based hierarchy is independently represented by a set of anchor splats, as shown in Fig. 2. The lack of correlation between levels can hinder efficient progressive transmission and limit flexible detail adjustment across different parts of the scene, such as foveated rendering or distance-aware rendering [15, 23, 25].
- Furthermore, since different levels of detail do not share content, each level needs to redundantly encode and represent similar visual information using separate sets of Gaussian splats, significantly enlarging both the global and intermediate model size, as shown in Figure 1.

In this paper, we propose LapisGS[1], a layered representation for progressive streaming and rendering of 3DGS content. Inspired by scalable coding [1, 24, 31] and progressive LOD representation [15, 23, 24, 33, 38], this method is designed to efficiently stream and render photo-realistic 3D objects and scenes by leveraging a progressive framework that enables dynamic adaptation to varying levels of detail. As shown in Fig. 2, at the core of our approach is a layered structure for cumulative representation, where each layer adds additional details to the existing base layers, progressively refining the representation. To force coherence among layers while avoiding re-encoding lower layers, we incorporate dynamic opacity optimization during training, allowing for selective adjustment of layer contributions for optimal visual fidelity. We then utilize an occupancy map to track and exclude less important Gaussian splats during streaming and rendering, improving computational and storage efficiency.

Our contributions can be summarized as follows.

- A progressive layered approach for 3DGS encoding multiple levels of detail into a single-layered model, supporting adaptive streaming and seamless rendering.
- Dynamic opacity optimization and management, which ensures consistency across varying resolution levels but also adjusts layer contributions selectively to maintain visual fidelity. As a result, our approach reduces data size and additionally enhances computational efficiency by managing splats dynamically with occupancy maps.
- Flexible and adaptive rendering, which enables seamless transitions and view-adaptive rendering strategies without the need for separate models for each level of detail.
- Extensive experiments on diverse 3D contents demonstrate the effectiveness of our method, achieving high-quality rendering and low resource cost, with up to 50.71% improvement in SSIM, 286.53% improvement in LPIPS with 23% of the original model size.

By drawing parallels to scalable coding and LOD representation, our progressive representation effectively balances the need for high-quality rendering with bandwidth-aware streaming, providing a robust solution for adaptive 3D content delivery.

To foster collaboration and further research, we will release our source code to the research community and make our pre-trained layered 3DGS models publicly available.

2. Background and Related Work

2.1. 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) [11] is a method used for real-time photorealistic radiance field rendering by representing a 3D scene as a collection of 3D Gaussian splats.

[1] *Lapis* means "layer" in Malay, the national language of Singapore — the host of 3DV'25. The logo in the title depicts *kuih lapis*, or "layered cake", a local delight in Singapore and neighboring countries

Each 3D Gaussian splat is characterized by a set of attributes, including its position $\mathbf{x}$, a covariance matrix $\mathbf{\Sigma}$ decomposed by a scaling matrix $\mathbf{S}$ and a rotation matrix $\mathbf{R}$, opacity σ, and view-dependent color c represented by a set of Spherical Harmonic (SH) coefficients. Specifically, each 3D Gaussian is defined as $G(\mathbf{x}) = exp(-\frac{1}{2}(\mathbf{x})^T\mathbf{\Sigma}^{-1}(\mathbf{x}))$. In the rendering stage [39], each 3D Gaussian is projected into 2D camera coordinates, denoted as $G'(\mathbf{x}')$. To compute the color of a pixel $\mathbf{x}'$, a tile-based rasterizer is employed to sort the projected splats in front-to-back depth order and blend their colors with α-blending. The α-blending weights are deprived as $\sigma G'(\mathbf{x}')$.

These Gaussian splats are derived from a sparse Structure-from-Motion (SfM) point cloud and are refined through a gradient-descent-based optimization process interleaved with adaptive refinements.

2.2. Level-of-Detail 3DGS

Level-of-details (LODs) are fundamental to scalable rendering solutions, allowing for the adjustment of detail levels based on computational resources and user needs. While LODs have been explored for point cloud representations, applying them to 3DGS presents unique challenges.

Recent efforts have been made to build multiscale representations for 3DGS [4, 12, 16, 17, 22, 34]. For instance, Yan *et al.* [34] proposed to add larger and coarser Gaussian splats for lower resolutions by aggregating the smaller and finer Gaussians from higher resolutions, creating independent Gaussian splats layers at different scales to mitigate the aliasing artifacts during rendering. To reconstruct and render large-scale scenes with 3DGS, Liu *et al.* [16] divided the whole scene into spatially adjacent blocks and trained each block in parallel. Within each block, different LODs were created with LightGaussian [7], by pruning and post-optimizing the splats. Kerbl *et al.* [12] proposed a tree-based hierarchy designed for real-time rendering of large-scale scenes. Their approach involves dividing the scene into chunks and constructing a tree for each chunk, where both interior and leaf nodes are represented by Gaussian splats. Interior nodes are formed by merging child splats and then optimized at each level separately, while leaf nodes originate directly from the initial optimization process. To ensure smooth transitions between levels, splat interpolation is employed. Similarly, Ren *et al.* [22] use an octree structure to partition the 3D space, with each octree level corresponding to a set of anchor Gaussians that define the LOD. They incorporate Scaffold-GS [17], which leverages neural Gaussians and MLPs to predict anchor-wise features, allowing for a more compact representation. Linear interpolation of rendered 2D images is used to achieve smooth transitions between levels.

However, these existing methods are primarily designed for reconstruction quality and rendering speed, without considering the challenges of adaptive streaming, as we discussed in Sec. 1. Space-based hierarchies in 3DGS feature discrete quality layers, each independently represented by a set of anchor splats, as shown in Fig. 2. This approach requires substantial computational resources for maintenance and navigation, especially with frequent level transitions, and limits view-adaptive rendering. In contrast, our model supports progressive streaming and flexible detail adjustment, reducing storage needs while enabling smoother level transitions and effective view-adaptive rendering.

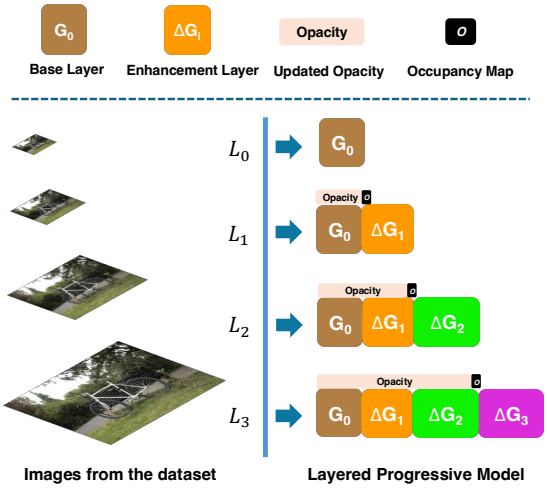


Figure 3. The overview of LapisGS. The framework progressively constructs a layered model, starting from the low-resolution base layer (L_0) and adding higher-resolution enhancement layers (L_1, L_2, L_3). Dynamic opacity optimization and occupancy maps are employed to refine splat contributions and optimize data size.

3. Methodology

Our method LapisGS is based on training a 3DGS model at successive level of progressively higher resolution to create a multiscale representation. Initially, a low-resolution dataset is used to establish a base layer. As the training progresses, new enhancement layers are added, each trained on incrementally higher resolution versions of the dataset. These layers build upon and refine the details captured in the previous layers. While the parameters of the prior layers remain fixed, their opacity is optimized to adjust the influence of each layer dynamically. We also utilize occupancy maps to track the contributions of splats. During streaming and rendering, these transparent splats are excluded, which reduces the overall model size and improves computational efficiency. To ensure smooth transitions between resolution levels, we employ interpolation of opacity values between adjacent layers. Fig. 3 shows the overview of LapisGS.

(a) $\mathbf{G}_1$. (b) $\Delta\mathbf{G}_2$. (c) $\Delta\mathbf{G}_3$. (d) $\mathbf{G}_3$.

Figure 4. Sample renderings of *hotdog*. The enhancement layers $\Delta\mathbf{G}_2$ and $\Delta\mathbf{G}_3$ capture higher frequency features and can be iteratively added to the layer $\mathbf{G}_1$ to obtain the model $\mathbf{G}_3$.

3.1. Layered Progressive 3DGS

Layered Progressive Representation. We denote $N+1$ as the total number of quality levels for the layered 3DGS model, which also corresponds to the total number of training stages in the progressive training scheme. We can then denote L_i as the i-th level of detail where $i \in \{0, 1, \ldots, N\}$. Given a full-resolution multi-view image set $\mathbf{D}_N$, we can build an image pyramid $\{\mathbf{D}_i\}_{i=0}^{N}$:

$$\mathbf{D}_i = \left\{ \left(V_m^i, X_m^i \right) \right\}_{m=1}^{M}, \tag{1}$$

where V_m^i is the camera matrix, X_m^i is the corresponding image, and M is the number of views.

Starting from the lowest quality level, we initially optimize a set of 3D Gaussian splats, denoted as $\mathbf{G}_0$. As training advances, views with higher resolution are considered at each layer. Consequently, the Gaussian splats at various quality levels are represented as $\{\mathbf{G}_i\}_{i=0}^{N}$.

As discussed in Sec. 1, the core idea is to create a layered structure comprising a base layer G_0 and enhancement layers. As training progresses to a higher quality level, new enhancement layers are optimized and integrated with prior layers. Formally, we can represent the layered progressive model as $\{\mathbf{G}_0, \{\Delta\mathbf{G}_i\}_{i=1}^{N}\}$, where

$$\mathbf{G}_i = \mathbf{G}_0 + \sum_{k=1}^{i} \Delta\mathbf{G}_\mathbf{k}, i \in \{1, 2, \ldots, N\}, \tag{2}$$

where $\Delta\mathbf{G}_\mathbf{k}$ is the k-th enhancement layer.

In this layered structure and progressive training scheme, a rough scene layout is constructed, allowing lower-frequency features to be learned in the early stages. This layout serves as a foundation for higher quality levels in subsequent training stages. By building upon information from previous levels, the model can focus more on capturing higher frequency features, thereby speeding up convergence and reducing redundancy across different quality levels. Fig. 4 shows an example of the above process.

Multi-level Optimization. During the training of each level in LapisGS, we focus on optimizing the enhancement layer $\Delta\mathbf{G}_i$ while maintaining the parameters of the preceding layers $\{\mathbf{G}_0, \Delta\mathbf{G}_1, \ldots, \Delta\mathbf{G}_{i-1}\}$ as fixed, except for their opacity values.

The decision to optimize only the opacity values of the previous layers, rather than other parameters, is based on the balance between efficient layer integration and visual coherence. 3DGS applies standard α-blending for rendering, indicating that the contribution of each splat is typically additive and weighted by its opacity. Besides, unlike other attributes of Gaussian splats (such as position or scale), changing opacity does not alter the spatial information of the scene [6, 7, 22, 29]. Therefore, opacity optimization allows for fine-tuning the visibility and influence of Gaussian splats from earlier layers without altering their foundational features and changing the layered structure. In other words, this selective optimization strategy enables the model to refine the contributions of existing layers, ensuring that previously learned features are kept, while the current enhancement layer focuses on capturing new details and improving the overall representation.

For each level L_i, the optimization process is driven by minimizing a rendering loss function $\mathcal{L}_{\Delta\mathbf{G}_i}$, which consists of two components: an L_1-norm loss $\mathcal{L}_1$ and a D-SSIM loss $\mathcal{L}_\mathrm{D}$. The rendering loss is defined as follows:

$$\mathcal{L}_{\Delta\mathbf{G}_i} = \lambda \sum_{m=1}^{M} \mathcal{L}_1(\hat{X}_m^i, X_m^i) + (1-\lambda) \sum_{m=1}^{M} \mathcal{L}_\mathrm{D}(\hat{X}_m^i, X_m^i), \tag{3}$$

where X_m^i represents the ground truth image for view m at the i-th quality level, and $\hat{X}_m^i$ is the image rendered of $\mathbf{G}_i$, using all enhancement layers up to L_i.

Notably, $\mathcal{L}_1$ loss estimates perceived errors and is not sensitive to blurriness or low-resolution artifacts that do not alter the image's structure [32, 37]. In our progressive training pipeline, this insensitivity can result in premature convergence, where the model fails to update and densify the "low-layer" Gaussian splats. Existing works [3, 34, 35, 37] offer sophisticated solutions. For example, Zhang *et al*. [37] propose to utilize frequency information to regularize the Gaussian densification, by incorporating frequency term in the loss function. Nevertheless, addressing this problem is orthogonal to our work. In our approach, we adopt a straightforward step to alleviate the problem by giving more weight to the SSIM loss with $\lambda = 0.2$, to ensure that the model prioritizes maintaining structural integrity.

By minimizing the rendering loss at each level, the model incrementally refines its representation, effectively balancing the integration of new information and the learned features of lower-resolution layers. This approach ensures that the final model achieves high visual fidelity and efficient splat coding across successive quality levels.

Representation Compaction and Adjustment. Gaussian splats from the first layers capture broader and low-frequency features essential for constructing a rough scene layout. As the training process progresses to higher resolutions, the model focuses on capturing fine-grained details

Figure 5. Example of the smooth level transition on *Lego*.

that the lower-resolution splats cannot effectively represent. Consequently, the opacity of these "low-layer" splats is reduced during the optimization of "high-layer" splats, indicating their decreased importance in the rendering process.

To improve transmission and computational efficiency, we introduce an occupancy map O_i for each level L_i in our layered 3DGS model. This map tracks the opacity of individual Gaussian splats, identifying those that fall below a specified opacity threshold. These splats are marked as less important and can be omitted during both streaming and rendering processes, significantly reducing the model size during transmission and rendering. We set the opacity threshold at 0.005, following the default pruning phase setting from the original 3DGS work [11].

Nevertheless, this approach could allow for adaptive bitrate streaming by dynamically selecting splats based on their opacity. By adjusting the opacity threshold, the system could fine-tune the amount of data transmitted, adapting to varying network bandwidths and device capabilities [7, 29].

3.2. Continuous level transition

A critical challenge in LOD rendering is achieving seamless visual transitions between different resolution levels. Abrupt changes between levels can lead to visual artifacts, degrading the user experience, especially in scenarios with limited bandwidth or dynamic view changes [29].

We address this by linearly interpolating the opacity of a given splat between adjacent levels. As discussed in Sec. 3.1 and Sec. 2.1, opacity leverages each splat's contribution during rendering, and by adjusting these weights, we effectively blend representations from different quality levels in a plausible manner. Interpolating the opacity allows for gradual detail blending without altering spatial information, resulting in smooth visual changes. Our method aligns with the additive nature of Gaussian splat rendering and leverages the human visual system's lower sensitivity to gradual intensity changes.

Specifically, for a target resolution r_t that falls between the resolutions of two adjacent quality levels L_i and L_{i+1}, we define a continuous interpolation factor t. This factor is calculated as

$$t(r_t) = \frac{r_t - r_i}{r_{i+1} - r_i}, \tag{4}$$

where r_i and r_{i+1} are the resolutions of quality levels L_i and L_{i+1}, respectively, and $r_i \leq r_t < r_{i+1}$.

Our interpolation scheme ensures smooth transitions between different levels of detail in our model. We illustrate the renderings of the interpolated models between L_0 and L_1 in Fig. 5.

Additionally, by dynamically and continuously adjusting the influence of splats based on their opacity, LapisGS could allow for efficient view-adaptive rendering, which optimizes rendering performance by adjusting the quality levels based on the viewer's gaze and the distance to objects. Detailed analysis and comparison, along with illustrative figures, can be found in the supplementary materials.

4. Experiments

In this section, we first detail the experimental setup of LapisGS in Section 4.1. We then evaluate the performance of our approach in Section 4.2. We also conduct ablation studies to analyze the contributions of key components within our method.

4.1. Experimental Setup

Dataset. We evaluated our method using 19 objects and real scenes from various datasets, including Synthetic Blender [18], Mip-NeRF360 [2], Tanks&Temples [13], and Deep Blending [8]. These datasets encompass a diverse range of object-centric, indoor, and outdoor scenes, providing a robust basis for testing. In addition to full-scale images, we down-scaled each dataset by factors of $2\times$, $4\times$, and $8\times$, reducing the image resolution by half, a quarter, and one-eighth in each direction, respectively. This provided multiple scales for constructing the layered representation.

Implementation. Our implementation is based on the official release of the 3D Gaussian Splatting code [10]. We initially trained the base layer from the dataset with the lowest resolution. In subsequent training stages, we fixed the parameters of Gaussian splats in prior layers, except for their opacity, and trained the enhancement layer and the opacity of prior layers on datasets with the corresponding quality levels. All hyperparameters remained consistent across training stages. The training process was conducted on an NVIDIA A100 GPU.

Table 1. Quantitative comparison results on synthetic Blender dataset [18] at different quality levels. The model size is normalized.

Method	L_0			L_1			L_2			L_3		
	SSIM ↑	LPIPS ↓	Size ↓	SSIM ↑	LPIPS ↓	Size ↓	SSIM ↑	LPIPS ↓	Size ↓	SSIM ↑	LPIPS ↓	Size ↓
Downsample	0.827	0.117	0.127	0.875	0.092	0.141	0.907	0.069	0.252	0.929	0.060	0.430
Single	0.747	0.131	1.000	0.864	0.070	1.000	0.946	0.032	1.000	0.969	0.027	1.000
Multiscale	**0.984**	**0.014**	**0.127**	0.980	0.016	0.272	**0.976**	**0.018**	0.540	0.969	**0.027**	1.000
LapisGS	**0.984**	**0.014**	**0.127**	**0.981**	**0.015**	**0.141**	0.970	0.026	**0.252**	**0.970**	0.044	**0.430**
Δ	-	-	-	0.001	-0.001	-92.91%	-0.006	0.008	-114.29%	0.001	0.017	-132.56%

Table 2. Quantitative comparison results on Mip-NeRF360 dataset [2] at different quality levels. The model size is normalized.

Method	L_0			L_1			L_2			L_3		
	SSIM ↑	LPIPS ↓	Size ↓	SSIM ↑	LPIPS ↓	Size ↓	SSIM ↑	LPIPS ↓	Size ↓	SSIM ↑	LPIPS ↓	Size ↓
Downsample	0.548	0.314	0.239	0.678	0.236	0.413	0.778	0.194	0.520	0.870	0.166	0.568
Single	0.635	0.201	1.000	0.752	0.164	1.000	0.881	0.119	1.000	0.918	0.124	1.000
Multiscale	**0.957**	**0.052**	**0.239**	**0.947**	**0.065**	0.531	**0.928**	**0.096**	0.783	0.918	**0.124**	1.000
LapisGS	**0.957**	**0.052**	**0.239**	0.936	0.080	**0.413**	**0.928**	0.111	**0.520**	**0.925**	0.161	**0.568**
Δ	-	-	-	-0.011	0.015	-28.57%	0.000	0.015	-50.58%	0.007	0.037	-76.06%

Table 3. Quantitative comparison results on Tank&Temples dataset [13] at different quality levels. The model size is normalized.

Method	L_0			L_1			L_2			L_3		
	SSIM ↑	LPIPS ↓	Size ↓	SSIM ↑	LPIPS ↓	Size ↓	SSIM ↑	LPIPS ↓	Size ↓	SSIM ↑	LPIPS ↓	Size ↓
Downsample	0.530	0.340	0.104	0.602	0.302	0.241	0.724	0.238	0.424	0.868	0.154	0.608
Single	0.640	0.198	1.000	0.764	0.140	1.000	0.885	0.092	1.000	0.923	0.106	1.000
Multiscale	**0.958**	**0.051**	**0.104**	**0.946**	**0.060**	0.272	0.934	**0.077**	0.528	**0.923**	**0.106**	1.000
LapisGS	**0.958**	**0.051**	**0.104**	0.942	0.076	**0.241**	**0.935**	0.103	**0.424**	0.916	0.163	**0.608**
Δ	-	-	-	-0.004	0.016	-12.86%	0.001	0.026	-24.53%	-0.007	0.057	-64.47%

Table 4. Quantitative comparison results on Deep Blending dataset [8] at different quality levels. The model size is normalized.

Method	L_0			L_1			L_2			L_3		
	SSIM ↑	LPIPS ↓	Size ↓	SSIM ↑	LPIPS ↓	Size ↓	SSIM ↑	LPIPS ↓	Size ↓	SSIM ↑	LPIPS ↓	Size ↓
Downsample	0.820	0.168	0.319	0.880	0.132	0.431	0.909	0.157	0.489	0.913	0.236	0.544
Single	0.829	0.152	1.000	0.910	0.095	1.000	0.959	0.094	1.000	0.959	0.171	1.000
Multiscale	**0.966**	**0.045**	**0.319**	**0.967**	**0.055**	0.611	**0.967**	**0.077**	0.825	**0.959**	**0.171**	1.000
LapisGS	**0.966**	**0.045**	**0.319**	0.961	0.097	**0.431**	0.954	0.125	**0.489**	0.951	0.219	**0.544**
Δ	-	-	-	-0.006	0.042	-41.76%	-0.013	0.048	-68.71%	-0.008	0.048	-83.82%

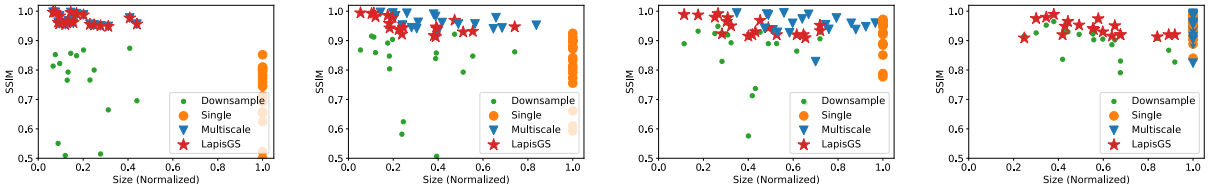
Comparison Method. We compare LapisGS with three alternative approaches:

- *Single*. A full-resolution model is trained and then rendered at multiple scales. This approach demonstrates the trade-offs in scalability and quality when a single model is tasked with adapting to different resolutions without specific optimization for each scale.
- *Multiscale*. Separate models are trained independently for each scale ($1\times, 2\times, 4\times$, and $8\times$), resulting in four distinct models that form a multiscale representation. This is similar to tree-based hierarchies where each level is independently represented shown in Fig. 2, and serves as the upper bound for reconstruction quality.
- *Downsample*. Following Fan *et al*. [7], this method downsamples the full-resolution model by calculating significance scores based on the opacity and scale of each splat, removing those with lower scores. This approach serves as the lower bound and highlights the limitations of using traditional point cloud down- and up-sampling techniques [26, 36] to construct an LOD representation for 3DGS which is a point-based representation.

Metrics. We evaluate the visual quality using SSIM and LPIPS. We exclude PSNR from our evaluation as it primarily estimates pixel-wise error, which makes it less sensitive to blurriness and low-resolution artifacts [32, 37], as discussed in Sec. 3.1. In addition to quality, we compare the model sizes across different methods. Given that 3DGS models vary significantly in size depending on the complexity of the 3D content they represent, we normalize the model sizes to a range of $(0, 1]$ by dividing each model's size by the maximum model size within that 3D content.

4.2. Results and Evaluation

Quantitative Comparison. We present the results from four datasets in Tabs. 1 to 4. As our comparison focuses on the trade-off between model size and rendering quality, the size and quality difference and change in size of our method compared to the Multiscale method is also presented. Note that we quantify the size change relative to the new model size, which expresses the reduction as a negative percentage. A more negative value indicates a greater size decrease. To better illustrate the comparison, we also show



(a) L_0. (b) L_1. (c) L_2. (d) L_3.

Figure 6. Each point represents the overall quality of a scene/object with the corresponding model size at a given quality level. As the level increases, LapisGS achieves high visual quality (SSIM) with a considerably smaller model size.

the visual quality against normalized model size for each scene and object at each resolution scale, in Fig. 6. Several key observations are highlighted.

First, the downsampled model shows significant degradation in visual quality as model size decreases. This suggests that traditional downsampling methods for point clouds are inadequate for 3DGS, primarily because they fail to re-optimize or re-learn the model for the reduced set of splats. Given the unique distribution and anisotropic nature of Gaussian splats, simply reducing the number of splats without retraining does not capture the high-frequency details or maintain the model's original fidelity.

Second, LapisGS consistently achieves the smallest model size across all datasets and scales, demonstrating the effectiveness of our progressive multiscale training pipeline: by enabling feature sharing across different LODs and multi-level optimization and occupancy maps, our approach significantly compacts the representation. This reduction in model size directly correlates to decreased rendering times, further illustrating the efficiency of our layered progressive model in adaptive rendering scenarios.

Third, LapisGS achieves notable improvements in both visual quality and model size compared to the single-scale model. At lower resolutions, our approach yields enhancements of up to 50.71% in SSIM, 286.53% in LPIPS with 23% of the original model size. These results emphasize the necessity of constructing a multiscale 3DGS LOD model and highlight the effectiveness and efficiency of our method in maintaining visual fidelity while reducing data overhead.

Finally, LapisGS rivals the multiscale model, which serves as the upper bound for reconstruction quality, achieving comparable visual fidelity with up to a 2.33× smaller model size. This demonstrates that our method effectively balances high and low resolutions, creating a compact yet highly detailed representation.

In summary, our layered progressive model ensures high visual quality with a compact model that supports efficient and adaptive rendering, making it an effective solution for adaptive 3DGS streaming.

Qualitative Comparison. We present the qualitative comparison with the other methods in Fig. 7. More detailed qualitative comparisons are shown in the supplementary materials due to the page limit. As observed, LapisGS successfully avoids the common artifacts seen in downsampled and single-scale models, achieving results comparable to the Multiscale method but with significantly reduced model size. This improvement highlights the effectiveness of our layered progressive model in capturing fine-grained features while maintaining a compact model size, making it an ideal choice for adaptive 3DGS streaming.

Table 5. The average size and quality difference in percentage of LapisGS at different scales, compared to the Freeze method.

Dataset	Δ SSIM ↑	Δ LPIPS ↓	Δ Size ↓
Synthetic [18]	0.82%	-26.74%	-118.90%
360 [2]	4.61%	-22.34%	-142.19%
T&T [13]	5.01%	-32.31%	-123.10%
DB [8]	1.71%	-12.28%	-201.46%

Ablation Study. To build a layered feature-sharing LOD model, we chose to optimize only the opacity values of the previous layers and freeze other parameters to balance efficient layer integration and visual coherence. This step empowers the representation compaction and adjustment, as well as view-adaptive rendering, by dynamically determining which splats contribute most significantly to the visual fidelity at various distances from the viewpoint.

To explore the effect of the proposed multi-level optimization approach, we conducted an ablation study. Specifically, we evaluated a variant of our method where all parameters, including opacity, are frozen when training enhancement layers. This method, denoted as "Freeze", serves as a baseline to highlight the importance of dynamic opacity optimization. We evaluated the visual quality and model size on all four datasets, comparing our method against the Freeze method.

We present the average results in Tab. 5 and per-scene quantitative and qualitative results in the supplementary. The results highlight that without opacity optimization, model size increases significantly, and visual quality deteriorates. The visual quality degrades because lower-layer splats, which cannot capture high-frequency features,

Figure 7. Sample renderings of *Garden* at different levels. LapisGS eliminates the usual artifacts seen in downsampled and single-scale models, achieving comparable quality to the Multiscale method with considerably lower data size, as shown in Tab. 2.

continue to influence higher-layer representations. Consequently, the model size increases significantly as additional splats are added to compensate for the deficiencies of these lower-layer splats, leading to redundancy. In contrast, our method achieves more compact models and superior visual fidelity by efficiently pruning less significant splats.

Overall, LapisGS ensures a balance between model size and rendering quality, demonstrating the effectiveness of dynamic opacity optimization for scalable 3DGS.

5. Discussion and Conclusion

In this paper, we introduced LapisGS, a layered progressive 3DGS designed for adaptive streaming. Our approach addresses key challenges in 3D content streaming by optimizing the balance between visual quality and model size. We leverage visual information sharing across multiple LODs, which is crucial for real-time rendering and adaptive streaming where resources are limited. A key advantage of LapisGS is its progressive nature, where each layer builds upon the previous one, allowing for efficient and coherent integration of details. By selectively optimizing the opacity of splats in lower layers, we reduce redundancy and ensure that only relevant splats enhance visual fidelity, particularly for high-frequency details. This progressive framework avoids the inefficiencies of traditional models that either downsample or train separate models for each scale. Our method demonstrates superior efficiency and flexibility, achieving substantial improvements in visual quality and model compactness, as evidenced by high SSIM and low LPIPS scores across various datasets. Additionally, it supports view-adaptive streaming and rendering through dynamic pruning and interpolation of opacity based on network conditions, ensuring smooth transitions between quality levels and a seamless user experience. For a complementary analysis and comparison, including visual illustrations, we direct readers to the supplementary materials.

While our method performs well, there are areas for improvement. First, our approach is optimized for static scenes. Extending the model to handle dynamic elements or real-time updates in changing environments is an important direction for future research, especially for more interactive applications. Second, a notable aspect not explored in this work is the evaluation of performance under fluctuating network conditions. These tasks are beyond the scope of this paper, which focuses on developing an adaptive streaming model. However, we refer readers to our recent work [30], where we develop, implement, and evaluate the first DASH-based dynamic 3DGS streaming system built on our LapisGS. This system demonstrates superior performance in both live and on-demand streaming.

References

[1] Jae-Kyun Ahn, Yeong Jun Koh, and Chang-Su Kim. Efficient fine-granular scalable coding of 3D mesh sequences. *IEEE Transactions on Multimedia*, 15(3):485–497, 2013. 2

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 5460–5469. IEEE, 2022. 5, 6, 7, 1, 4

[3] Lizhe Chen, Yan Hu, Yu Zhang, Yuyao Ge, Haoyu Zhang, and Xingquan Cai. Frequency-importance Gaussian splatting for real-time lightweight radiance field rendering. *Multimedia Tools and Applications*, 83(35):83377–83401, 2024. 4

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

[5] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. HAC: hash-grid assisted context for 3D Gaussian splatting compression. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part VII*, pages 422–438. Springer, 2024. 1

[6] Zilong Chen, Feng Wang, Yikai Wang, and Huaping Liu. Text-to-3D using Gaussian splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 21401–21412. IEEE, 2024. 4

[7] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. LightGaussian: Unbounded 3D Gaussian compression with 15x reduction and 200+ FPS. *CoRR*, abs/2311.17245, 2023. 1, 3, 4, 5, 6

[8] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel J. Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics*, 37(6):257, 2018. 5, 6, 7, 1, 2

[9] Peng Jiang, Gaurav Pandey, and Srikanth Saripalli. 3DGS-ReLoc: 3D Gaussian splatting for map representation and visual relocalization. *CoRR*, abs/2403.11367, 2024. 1

[10] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. https://github.com/graphdeco-inria/gaussian-splatting, 2023. 5

[11] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4):139:1–139:14, 2023. 1, 2, 5

[12] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3D Gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics*, 43(4):62:1–62:15, 2024. 2, 3, 1

[13] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4):78:1–78:13, 2017. 5, 6, 7, 1, 3

[14] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3D Gaussian representation for radiance field. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 21719–21728. IEEE, 2024. 1

[15] David Li and Amitabh Varshney. Progressive multi-scale light field networks. In *International Conference on 3D Vision, 3DV 2022, Prague, Czech Republic, September 12-16, 2022*, pages 231–241. IEEE, 2022. 2

[16] Yang Liu, Chuanchen Luo, Lue Fan, Naiyan Wang, Junran Peng, and Zhaoxiang Zhang. CityGaussian: Real-time high-quality large-scale scene rendering with Gaussians. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XVI*, pages 265–282. Springer, 2024. 2, 3

[17] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-GS: Structured 3D Gaussians for view-adaptive rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 20654–20664. IEEE, 2024. 1, 3

[18] 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 *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, pages 405–421. Springer, 2020. 5, 6, 7

[19] Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3D scene representation via self-organizing Gaussian grids. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part LXXXV*, pages 18–34. Springer, 2024. 1

[20] K. L. Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Compact3D: Compressing Gaussian splat radiance field models with vector quantization. *CoRR*, abs/2311.18159, 2023.

[21] Panagiotis Papantonakis, Georgios Kopanas, Bernhard Kerbl, Alexandre Lanvin, and George Drettakis. Reducing the memory footprint of 3D Gaussian splatting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 7(1):16:1–16:17, 2024. 1

[22] 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. *CoRR*, abs/2403.17898, 2024. 2, 3, 4, 1

[23] Markus Schütz, Katharina Krösl, and Michael Wimmer. Real-time continuous level of detail rendering of point clouds. In *IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2019, Osaka, Japan, March 23-27, 2019*, pages 103–110. IEEE, 2019. 2, 1

[24] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Overview of the scalable video coding extension of the

H.264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, 2007. 2

[25] Yuang Shi and Wei Tsang Ooi. Perceptual impact of facial quality in MPEG V-PCC-encoded volumetric videos. In *Proceedings of the 16th International Workshop on Immersive Mixed and Virtual Environment Systems, MMVE 2024, Bari, Italy, April 15-18, 2024*, pages 71–74. ACM, 2024. 2

[26] Yuang Shi, Pranav Venkatram, Yifan Ding, and Wei Tsang Ooi. Enabling low bit-rate MPEG V-PCC-encoded volumetric video streaming with 3D sub-sampling. In *Proceedings of the 14th Conference on ACM Multimedia Systems, MMSys 2023, Vancouver, BC, Canada, June 7-10, 2023*, pages 108–118. ACM, 2023. 6

[27] Yuang Shi, Bennett Clement, and Wei Tsang Ooi. QV4: QoE-based viewpoint-aware V-PCC-encoded volumetric video streaming. In *Proceedings of the 15th ACM Multimedia Systems Conference, MMSys 2024, Bari, Italy, April 15-18, 2024*, pages 144–154. ACM, 2024. 1

[28] Yuang Shi, Simone Gasparini, Géraldine Morin, Chenggang Yang, and Wei Tsang Ooi. Sketch and Patch: Efficient 3D Gaussian representation for man-made scenes. *CoRR*, abs/2501.13045, 2025. 1

[29] Yuan-Chun Sun, Yuang Shi, Wei Tsang Ooi, Chun-Ying Huang, and Cheng-Hsin Hsu. Multi-frame bitrate allocation of dynamic 3D Gaussian splatting streaming over dynamic networks. In *Proceedings of the 2024 SIGCOMM Workshop on Emerging Multimedia Systems, EMS 2024, Sydney, NSW, Australia, August 4-8, 2024*, pages 1–7. ACM, 2024. 4, 5

[30] Yuan-Chun Sun, Yuang Shi, Cheng-Tse Lee, Mufeng Zhu, Wei Tsang Ooi, Yao Liu, Chun-Ying Huang, and Cheng-Hsin Hsu. LTS: A DASH streaming system for dynamic multi-layer 3D Gaussian splatting scenes. In *Proceedings of the 16th ACM Multimedia Systems Conference, MMSys 2025, Stellenbosch, South Africa, March 31-April 4, 2025*. ACM, 2025. 8

[31] Jianqiang Wang, Dandan Ding, Zhu Li, and Zhan Ma. Multi-scale point cloud geometry compression. In *31st Data Compression Conference, DCC 2021, Snowbird, UT, USA, March 23-26, 2021*, pages 73–82. IEEE, 2021. 2

[32] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 4, 6

[33] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. BungeeNeRF: Progressive neural radiance field for extreme multi-scale scene rendering. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXII*, pages 106–122. Springer, 2022. 2

[34] Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. Multi-scale 3D Gaussian splatting for anti-aliased rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 20923–20931. IEEE, 2024. 3, 4

[35] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-Splatting: Alias-free 3D Gaussian splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 19447–19456. IEEE, 2024. 4

[36] Anlan Zhang, Chendong Wang, Bo Han, and Feng Qian. YuZu: Neural-enhanced volumetric video streaming. In *19th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2022, Renton, WA, USA, April 4-6, 2022*, pages 137–154. USENIX Association, 2022. 6

[37] Jiahui Zhang, Fangneng Zhan, Muyu Xu, Shijian Lu, and Eric P. Xing. FreGS: 3D Gaussian splatting with progressive frequency regularization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 21424–21433. IEEE, 2024. 4, 6

[38] Junyu Zhu, Hao Zhu, Qi Zhang, Fang Zhu, Zhan Ma, and Xun Cao. Pyramid NeRF: Frequency guided fast radiance field optimization. *International Journal of Computer Vision*, 131(10):2649–2664, 2023. 2

[39] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus H. Gross. EWA volume splatting. In *12th IEEE Visualization Conference, IEEE Vis 2001, San Diego, CA, USA, October 24-26, 2001, Proceedings*, pages 29–36. IEEE Computer Society, 2001. 3