This CVPR Findings paper is the Open Access version, provided by the Computer Vision Foundation. Except for this watermark, it is identical to the accepted version; the final published version of the proceedings is available on IEEE Xplore.

Splatwizard: A Benchmark Toolkit for 3D Gaussian Splatting Compression

Xiang Liu[1,2] Yimin Zhou[1] Jinxiang Wang[1] Yujun Huang[1] Shuzhao Xie[1] Shiyu Qin[1]
Mingyao Hong[2] Jiawei Li[4] Yaowei Wang[2,3,*] Zhi Wang[1] Shu-Tao Xia[1] Bin Chen[3,*]
[1]Tsinghua University [2]Pengcheng Laboratory
[3] Harbin Institute of Technology, Shenzhen [4]Huawei Technologies Ltd.

Abstract

The recent advent of 3D Gaussian Splatting (3DGS) has marked a significant breakthrough in real-time novel view synthesis. However, the rapid proliferation of 3DGS-based algorithms has created a pressing need for standardized and comprehensive evaluation tools, especially for compression tasks. Existing benchmarks often lack the metrics needed to holistically assess the unique characteristics of different methods, such as rendering speed, rate-distortion trade-offs, memory efficiency, and geometric accuracy. To address this gap, we introduce Splatwizard, a unified benchmark toolkit designed specifically for 3DGS compression. Splatwizard provides an easy-to-use framework for benchmarking 3DGS compression models and reusing state-of-the-art techniques from prior work. In addition, it includes an integrated pipeline that automates the computation of key performance indicators, including image-based quality metrics, Chamfer distance on reconstructed meshes, rendering frame rates, and computational resource consumption. Code is available on https://github.com/splatwizard/splatwizard.

1. Introduction

With the rapid progress of computer vision, 3D reconstruction has emerged as a core research area, underpinning downstream applications in virtual reality and embodied intelligence. Among various 3D representations, 3D Gaussian Splatting (3DGS) [23] achieves state-of-the-art rendering quality and real-time performance, enabling a wide range of practical uses.

However, raw 3DGS scenes contain substantial redundancy. A 3DGS represents a scene with millions of Gaussian ellipsoids distributed in 3D space, whose projected splats jointly capture geometry and appearance. In practice, a single scene often requires up to 10^7 Gaussians, leading to hundreds of megabytes of storage and limiting the deployability of uncompressed 3DGS in real applications. Similar to the need for information compression in the image and video [3, 27, 32, 38, 39, 46, 47], this has driven the development of numerous 3DGS compression techniques, as summarized in Table 1. These methods introduce heterogeneous designs across preprocessing, rendering pipelines, and post-processing stages, resulting in large differences in code structure, system behavior, and implementation logic. Consequently, they exhibit notable trade-offs in FPS, memory footprint, model size, and reconstruction quality, making it difficult to achieve consistent evaluation or identify universally effective improvements.

To address this issue, several works [2, 53] have attempted to summarize or benchmark existing 3DGS compression methods. However, these efforts largely rely on reported results or cover only a subset of representative techniques. In contrast, Splatwizard is designed as a dedicated benchmark toolkit for 3DGS compression, with unified training, encoding, decoding, and evaluation protocols. This distinction is important in practice because downstream tasks prioritize different properties. For instance, streaming systems [14, 42, 58] demand extremely low encoding and decoding latency, while robotics applications [16, 28] require high inference throughput and minimal memory usage for onboard deployment.

Besides, another major limitation of existing evaluations is the absence of geometric assessment. Geometric reconstruction is a central problem in computer graphics, and numerous recent works [6, 15, 20, 21] focus on improving the surface reconstruction quality of 3DGS, drawing substantial attention from the community. However, research on 3DGS compression has so far concentrated almost exclusively on photorealistic metrics such as PSNR and SSIM. Although these metrics capture appearance fidelity, they overlook the fact that 3DGS is fundamentally a three-dimensional representation. As a result, existing evaluations fail to reveal how compression affects the geometric integrity of the underlying 3D structure.

To mitigate the aforementioned issues, we have developed a unified training and evaluation framework with three

*Corresponding author

Method	Year	Std Rasterizer	QAT	Entropy Model	Post Pruning	Compression
3DGS [23]	2023	✓	-	-	✗	✗
Trimming the Fat [1]	2024	✓	-	-	✓	✗
PUP-3DGS [18]	2025	✓	-	-	✓	✗
Speedy-Splat [17]	2025	✓	-	-	✗	✗
LightGaussian [11]	2024	✓	-	-	✓	✓
MesonGS [49]	2024	✓	-	-	✓	✓
C3DGS [36]	2024	✗	Fake	-	✓	✓
Compact3DGS [26]	2024	✗	RVQ	-	✗	✓
HAC [8]	2024	✗	Noise	Gaus	✗	✓
CAT-3DGS [56]	2025	✗	Noise	Gaus/Laplace	✗	✓

Table 1. Comparison of representative 3DGS compression methods. "Std Rasterizer" indicates whether the method uses the standard 3DGS rasterizer without introducing a customized renderer. "QAT" denotes the quantization-aware training strategy: "Fake" refers to fake quantization, "RVQ" to residual vector quantization, and "Noise" to noise-based quantization relaxation. "Entropy Model" lists the assumed latent distribution used for entropy coding, where "Gaus" and "Laplace" denote Gaussian and Laplace distributions, respectively.

defining properties: **1) Standardization**. It defines a unified pipeline encompassing data loading, model training, compression, and rendering evaluation, supporting reproducible comparison and rapid development of new methods. **2) Decoupling**. We decouple components such as rasterizers, entropy modules, codecs, and training schedules as independent modules. This scheduler-based design simplifies the integration of heterogeneous methods and supports flexible composition of existing components. **3) More Metrics**. Based on the same evaluation pipeline, we include not only photometric quality but also Gaussian count, runtime metrics, and geometric reconstruction accuracy, providing a more comprehensive benchmark.

In summary, our contributions are threefold:

- We propose a unified benchmark toolkit for 3DGS compression, with a scheduler-based training abstraction, decoupled modules, and a standardized evaluation pipeline.
- We reproduce existing 3DGS compression methods under the proposed framework and provide a fair benchmark with photometric, computational, and geometric metrics.
- We present ChimeraGS as a case study built on Splatwizard, demonstrating that the framework enables rapid composition of competitive compression pipelines.

2. Related Work

The high rendering quality and flexibility of 3DGS have motivated research aimed at improving both reconstruction fidelity and storage efficiency. In this section, we review related works on 3DGS and the methods proposed to compress and optimize the representation.

2.1. Novel View Synthesis

Novel view synthesis is a classic problem in graphics research. Many works on the topic focused on image-based rendering [13, 43] and Neural Radiance Fields [5, 35]. In recent years, 3D Gaussian Splatting (3DGS) [23] has emerged as an efficient and expressive representation for novel view synthesis and 3D reconstruction. By explicitly modeling scenes with anisotropic Gaussians, 3DGS achieves high-quality rendering and real-time performance, attracting extensive research attention.

Numerous works have improved the rendering pipeline through adaptive density control [17], anti-aliasing [55] and more realistic shading mechanisms [52]. These advances consolidate 3DGS as a versatile representation bridging explicit geometry and neural rendering. Our framework supports various static Gaussian Splatting pipelines with distinct models and optimization objectives, providing a unified platform for evaluation and analysis.

2.2. Compression of 3DGS

Despite its rendering efficiency, 3DGS remains storage-intensive, which has motivated a fast-growing line of research on compact Gaussian representations.

Pruning-based methods [1, 11, 17, 18, 50, 57] remove redundant or visually insignificant Gaussians by analyzing opacity, gradient importance, or visibility contribution, effectively reducing both model size and rendering cost. Quantization approaches [11, 36, 49, 50, 56] compress Gaussian attributes through scalar quantization, vector quantization, or learned codebooks, helping preserve high-frequency details under low bitrates. Distillation strategies [18, 56] transfer knowledge from a high-quality teacher model to a compact student representation. Entropy-based compression [8, 36, 56] further reduces storage through learned probabilistic coding of Gaussian parameters or by applying rate–distortion optimization to balance bit usage and rendering accuracy. Some works also generalize 3DGS compression to a wider range of information modalities [31, 60].

Although these studies have substantially advanced 3DGS compression, their implementations are often incon-

Feature / Tool	GSCodec	GauStudio	Ours
Evaluation Metrics	P	P	P/G/C
Rasterizer Num	1	6	12
Compression	Yes	No	Yes
Model Num	5	6	12

Table 2. Key differences among Splatwizard and others. Ours provides **P**hotometrical, **G**eometrical and **C**omputational evaluation on more models, while others only provide limited results. Our framework also offers the most rasterizers (Rasterizer Num) and compression models (Model Num). Note not all included rasterizers are used in the experiments of the paper.

sistent, making cross-method evaluation difficult. To address this, our work introduces a unified framework that integrates the aforementioned techniques under a consistent training and evaluation pipeline, facilitating reproducible and comprehensive benchmarking for future research.

2.3. Benchmark and Tools

Unlike fully neural network-based models, the 3DGS framework incorporates numerous components beyond conventional neural network architectures, with the rasterizer being a typical example. Furthermore, the inherently complex control flow of 3DGS has generated a need for substantial boilerplate code.

GSplat [54] serves as an open-source, efficient, and user-friendly library that provides an efficient implementation of 3D Gaussian Splatting. GauStudio [53] serves as a framework for modeling 3D Gaussian Splatting to provide standardized, plug-and-play components for users to easily customize and implement a 3DGS pipeline. GSCodec Studio [29] focuses specifically on 3DGS compression, offering a modular design for integrating and benchmarking different compression strategies such as pruning, quantization, and entropy coding. Beyond toolkits, 3DGS.zip [2] provides a comprehensive survey of 3DGS compression methods and compares their main characteristics. However, these studies also have limitations. 3DGS.zip [2] is a survey and does not provide a benchmark toolkit or reproduced results, while GSplat [54] mainly focuses on efficient rasterization and includes only one compression module.

Our work is also inspired by these studies and possesses unique characteristics. The most representative aspect is the design of a flexible modular mechanism, which integrates over 10 different rasterizers and other modules including entropy estimation and encoding. Furthermore, based on a scheduler mechanism, we have implemented a more streamlined template for Gaussian Splatting models. Table 2 shows some differences among mentioned methods. Using this foundation, we have reproduced several representative methods and conducted comprehensive performance comparisons.

3. Architecture

Fig. 1 summarizes the design of Splatwizard. Our goal is to provide a reusable benchmark infrastructure for 3DGS compression, so that new methods can be implemented under a shared training and evaluation protocol instead of ad hoc codebases. In the subsequent sections, we will provide a detailed introduction to the key mechanisms and functional modules of Splatwizard.

3.1. Dynamic and AOT Module Loading

3DGS research frequently requires customized rasterizers and CUDA/C++ extensions [11, 12, 21, 30]. In compression settings, entropy models and encoders [8, 56] further complicate dependencies and build procedures. To reduce this engineering burden, Splatwizard provides a unified CUDA/C++ integration interface based on PyTorch extensions [37], supporting both Just-in-Time loading and Ahead-of-Time compilation. This design lets users import modules through a unified Python interface while the framework handles compilation and dependency resolution.

Built on this mechanism, Splatwizard integrates multiple rasterizers and CUDA/C++ components used by existing compression methods. As a result, heterogeneous implementations can be evaluated under a consistent environment rather than isolated project setups.

3.2. Unified Pipeline

Beyond gradient updates, 3DGS training involves pruning, densification, quantization stages, and other iteration-dependent operations [23, 26]. Methods such as HAC [8] and CAT-3DGS [56] contain multi-stage procedures that are cumbersome to reproduce with scattered control logic. We therefore organize training into five stages: pre-scheduler execution, rendering, loss computation, post-scheduler execution, and optimizer update. A general flowchart can be referred to Fig. 1.

The scheduler executes registered tasks at designated iterations and automatically dispatches arguments according to task signatures [9]. This makes the training flow explicit and keeps implementations close to their original logic while remaining compatible with a unified benchmark interface, which significantly reduces the difficulty of integrating new methods into our framework. Listing 1 demonstrates the API for task registration in the scheduler. Table 3 shows the execution plan of vanilla 3DGS as a representative example. The execution plans for more reproduced methods are included in the Supplementary.

3.3. Function Library

Splatwizard groups reusable components into several libraries, as shown in Module Details columns of Fig. 1.

The Rasterizer Lib encapsulates rasterizers from standard 3DGS [23], accelerated rendering [33], reconstruction-

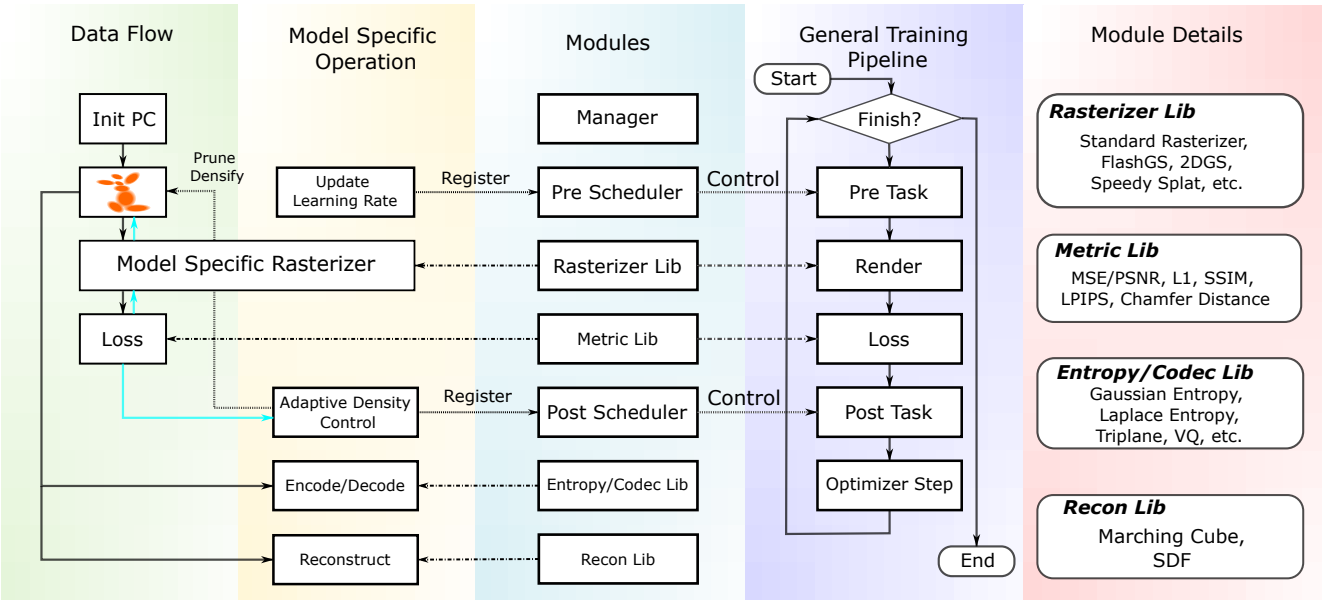


Figure 1. Framework of Splatwizard. We abstract Gaussian training into a unified pipeline with a scheduler for method-specific operations. Different rasterizers, codecs, and auxiliary modules can be plugged into different stages of the pipeline. The horizontal arrow lines in the diagram illustrate the modules used at different stages of Gaussian training. For instance, different models employ different adaptive density control mechanisms, and such control mechanisms can be registered into the scheduler.

Listing 1 Snippet used to register task.

```
class GaussianModel:
  def register_pre_task(
    self,
    scheduler: Scheduler,
    ppl: PipelineParams,
    opt: OptimizationParams
  ):
    scheduler.register_task(
      range(opt.iterations),
      task=self.update_learning_rate
    )
```

Stage	Task	Plan
Pre	update lr	range(30000)
	increase SH degree	range(0, 30000, 1000)
Post	collect gradient statistics	range(15000)
	prune and densify	range(500, 15000, 100)
	reset opacity	range(0, 15000, 3000)

Table 3. Schedule plan for vanilla 3DGS. We use python style range expression to illustrate detail execution time.

oriented variants [21, 24], and importance-aware pruning pipelines [10, 11, 17]. Different rasterizers can therefore be mixed within a single method when needed.

The Metrics Lib contains training losses and photometric metrics including PSNR, SSIM [44], and LPIPS [59]. It also includes geometric metrics, namely point-cloud and mesh-based Chamfer distance [10, 21], which are largely missing from prior compression benchmarks.

The Codec Lib collects modules for quantization, context modeling, and entropy coding. It covers representative choices used in prior work, including straight-through-estimator (STE) and noise-based quantization, vector quantization, hash-grid or triplane context models, and arithmetic/rANS/Huffman coding. Consolidating these modules makes it easier to analyze design trade-offs across methods.

The Reconstruction Lib converts optimized Gaussians into explicit geometry for evaluation. Following previous work, we render depth maps from training views and fuse them with TSDF using Open3D [61]. This module enables geometry-oriented benchmarking under the same framework as photometric evaluation.

3.4. Model Zoo

Based on the modular design above, we reproduce mainstream 3DGS compression methods within one codebase. Our reproduction principle is to preserve each method's core logic while factoring out shared functionality into common modules. This reduces engineering discrepancies and makes controlled comparison possible.

We also inspect reproduced implementations for issues that can affect correctness. A representative example is quantization boundary handling in entropy-coded models, where improper clamping may cause unstable decoding. By standardizing such components, Splatwizard serves both as

a benchmark and as a reliable implementation reference.

4. ChimeraGS

We use ChimeraGS as a case study to show how Splatwizard can compose modules from different methods into a single compression pipeline. ChimeraGS combines compact initialization, pruning, distillation, vector quantization, and accelerated rendering while remaining inside the same training and evaluation framework. Fig. 2 illustrates the framework of our method.

4.1. Pruning Strategy

The original LightGaussian [11] comprises multiple stages: pruning, distillation, and the final vector quantization encoding phase. Each of these stages has room for optimization. During the pruning stage, we can adopt more aggressive pruning methods and ratios to reduce the number of Gaussian primitives. For example, we can adopt the pruning strategy from PUP-3DGS [18] to significantly reduce the number of Gaussian kernels after pruning while maintaining comparable rendering quality. Additionally, beyond post-processing pruning, we can directly learn a compact Gaussian representation to replace the pruning stage. This approach often achieves higher compression rates.

We adopt Speedy-Splat [17] to obtain a highly compact initialization of Gaussians. Speedy-Splat utilizes a reparameterized approximation of the Hessian matrix to measure the importance of each Gaussian primitive:

$$\tilde{U}_i = \log |\nabla_{g_i} I_{\mathcal{G}} \nabla_{g_i} I_{\mathcal{G}}^T|, \tag{1}$$

where $I_{\mathcal{G}}$ is the rendered image. g_i is the 2D projected value of G_i at pixel p, given by

$$g_i = e^q, q = -\frac{1}{2}(p - \mu_{i_{2D}})\mathbf{\Sigma}_{i_{2D}}^{-1}(p - \mu_{i_{2D}})^T. \tag{2}$$

Since $\log$ is monotonically increasing, Equation 1 can be further rewritten as

$$\tilde{U}_i = (\nabla_{g_i} I_{\mathcal{G}})^2. \tag{3}$$

Based on this efficient importance evaluation metric, effective pruning was achieved during the training phase.

4.2. Distillation Guided SH Pruning

LightGaussian further reduces the number of parameters per Gaussian primitive by directly lowering the spherical harmonic (SH) degree and then applying distillation-based fine-tuning. This approach significantly reduces the encoded size. However, adopting a more aggressive strategy during the pruning stage may create a scenario in which simply reducing the order and then fine-tuning fails to adequately compensate for the quality loss caused by the degree reduction. Therefore, we adopt a dynamic pruning approach that determines, in a learnable manner, which Gaussian primitives undergo order reduction.

More specifically, we apply binary masks $M \in \{0,1\}^N$ to the degree-3 SH parameters of each Gaussian primitive:

$$\hat{\mathbf{c}}_n^{(3)} = M_n \mathbf{c}_n^{(3)}, \tag{4}$$

where n is the primitive index and $\mathbf{c}_n^{(3)}$ denotes the degree-3 SH parameter. During rendering, we use $\hat{\mathbf{c}}_n^{(3)}$ instead of $\mathbf{c}_n^{(3)}$.

Since the binary mask M is non-differentiable, we follow previous works [26] and use a straight-through estimator:

$$M_n = \mathrm{sg}(\mathbb{1}[\sigma(m_n) > c] - \sigma(m_n)) + \sigma(m_n), \tag{5}$$

where $\mathrm{sg}(\cdot)$ is the stop-gradient operator, $\mathbb{1}[\cdot]$ and $\sigma(\cdot)$ denote the indicator function and sigmoid function, respectively. To eliminate redundant degree-3 SH parameters, we introduce an additional loss:

$$\mathcal{L}_m = \frac{1}{N}\sum_{n=1}^{N} \sigma(m_n). \tag{6}$$

The use of learnable masks for pruning Gaussians or Gaussian parameters has also been explored in previous works [26, 41]. Our main improvement here is to leverage distillation to further enhance performance. Following LightGaussian [11], we use a teacher model to generate pseudo views for supervising the student model:

$$\mathcal{L}_{\text{distill}} = \frac{1}{HW} \left\| I_{\text{teacher}}(\mathbf{R}|\mathbf{t}) - I_{\text{student}}(\mathbf{R}|\mathbf{t}) \right\|_2^2, \tag{7}$$

where $I_{\text{teacher}}(\mathbf{R}|\mathbf{t})$ and $I_{\text{student}}(\mathbf{R}|\mathbf{t})$ are the rendered images of the teacher model and student model, respectively, given camera rotation $\mathbf{R}$ and position $\mathbf{t}$. The pseudo-view position is given by:

$$\mathbf{t}_{\text{pseudo}} = \mathbf{t}_{\text{train}} + \mathcal{N}(0, \sigma^2), \tag{8}$$

where $\mathbf{t}_{\text{train}}$ is the sampled training position, and $\mathcal{N}$ denotes a Gaussian distribution with mean 0 and variance σ^2. The final loss in the distillation stage is

$$\mathcal{L} = \mathcal{L}_{\text{distill}} + \lambda \mathcal{L}_m \tag{9}$$

By comprehensively applying learnable masks and knowledge distillation, our method can mitigate the performance loss caused by pruning, thereby improving the rate-distortion performance of the model.

4.3. Encoding and Rendering

In the encoding phase, we adopt the vector quantization encoder from LightGaussian. The main difference is that we use separate codebooks for the degree-2 and degree-3 SH parameters respectively. Since Gaussian rendering

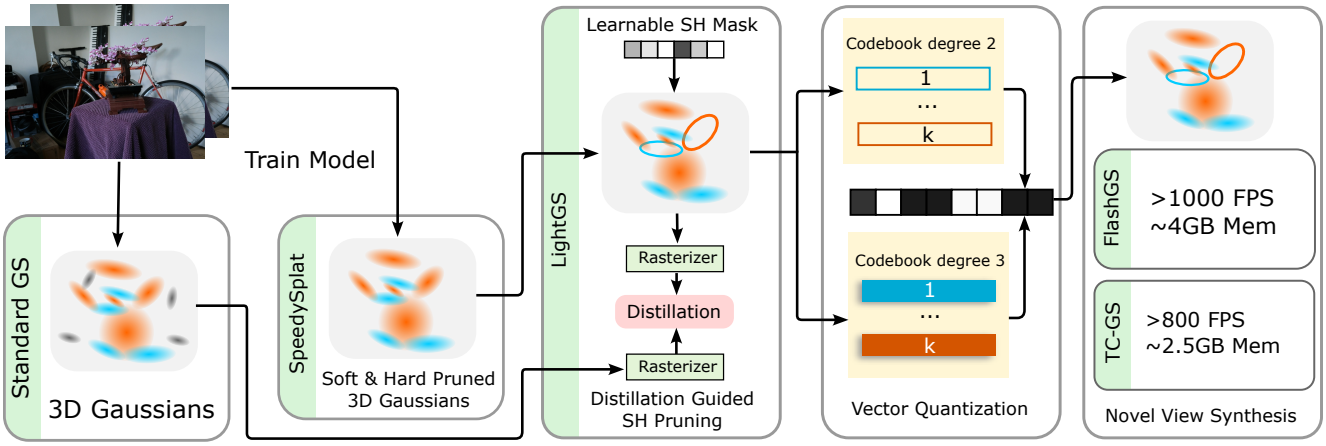


Figure 2. Framework of ChimeraGS. Since Splatwizard natively supports flexible module composition, we can easily combine multiple rasterizers within the same model. The labels on the left indicate the rasterizers used in each phase.

is position-order-independent, Gaussian points can be encoded separately according to their SH degrees. Additionally, to further improve encoding performance, we sort the Gaussian points using Morton order before the final encoding.

Since our method does not rely on a specially implemented rasterizer during the rendering phase, it can be easily integrated with existing rendering acceleration schemes. Here, we have experimented with FlashGS [12] and TC-GS [30] respectively. Combined with the model size advantage brought by compression itself, both methods have achieved impressive rendering speed.

5. Benchmarking

5.1. Benchmarking Setup

Our evaluation is conducted on six multi-view image datasets: Mip-NeRF 360 [4], Tanks&Temples [25], Deep Blending [19], BungeeNeRF [48], NeRF Synthetic [34] and DTU. Tanks&Temples features real-world scenes with complex geometry captured using laser scanners, while Mip-NeRF 360 contains 9 unbounded scenes enabling 360-degree rendering. Deep Blending provides diverse indoor and outdoor environments, BungeeNeRF focuses on extreme multi-scale scenarios, and NeRF Synthetic offers 8 photorealistic Blender-rendered scenes. For geometric evaluation, the DTU dataset [22] provides measurement-based ground-truth geometry. This enables us to evaluate the reconstruction accuracy of the model, thereby providing a more comprehensive reflection of its performance.

In terms of model selection, we evaluated all methods in the model zoo, with specific methods detailed in Table 1. All approaches were implemented or reproduced based on Splatwizard, using unified evaluation protocols and consistent training/testing dataset splits. Furthermore, to prevent potential data leakage from affecting bitrate evaluation, we strictly isolated the models used during the encoding and decoding phases. All included methods support encoding complete information into a single binary file, while the decoding phase utilizes only a newly initialized model and the corresponding binary file for testing.

5.2. Photorealistic Metrics Results

Fig. 3 presents evaluation results of all methods on Mip-NeRF 360, with results on additional datasets included in the Supplementary. Beyond the previously mentioned PSNR, SSIM, and LPIPS metrics, we also measured bitrate, Gaussian count, inference rendering speed, and peak memory usage during rendering. The figure reports cross-comparison results of these metrics.

From the conventional rate-distortion curves, we observe a trend: methods employing hybrid implicit neural networks and 3DGS generally achieve superior rate-distortion performance, while models that combine quantization operations with rate-distortion optimization also demonstrate relatively competitive performance. However, when considering Gaussian count as the metric for bitrate, the aforementioned observations may not necessarily hold.

Furthermore, comparisons of rendering frame rate and peak GPU memory usage provide a more comprehensive perspective. As expected, hybrid representation-based models generally exhibit lower frame rates compared to pruning-based approaches, though their impact on memory consumption is not significant.

The proposed ChimeraGS also achieves competitive results across several dimensions. Compared to prior approaches, our method attains rate-distortion performance closer to HAC [8], while achieving top-tier results in terms of Gaussian count. This supports our main claim that Splatwizard is not only useful for benchmarking, but also for rapidly prototyping hybrid pipelines by reusing modules from existing methods.

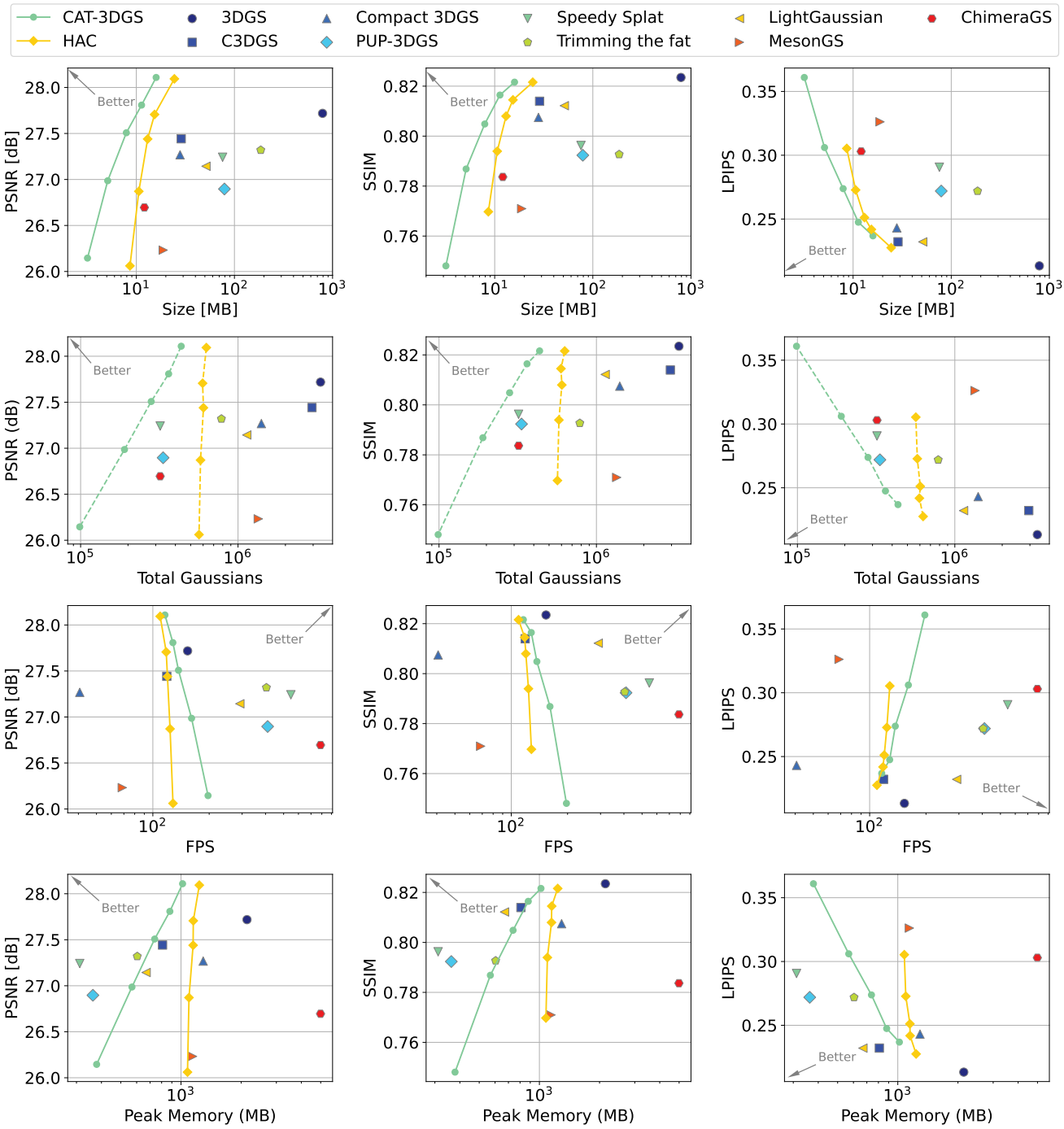


Figure 3. Results of Mip-NeRF 360 dataset. Note that when counting the total number of Gaussian primitives in HAC and CAT-3DGS, only the number of anchors is considered, and the values are not directly comparable to other methods, hence they are drawn with a dashed line.

5.3. Geometric Metrics Results

The geometric rate-distortion results on DTU are provided in Fig. 4. Unlike the previous section, photometric evaluation follows the standard novel-view-synthesis protocol, while geometric evaluation targets reconstruction fidelity and therefore uses the full dataset for training. The geometric curves should thus be interpreted as measuring scene fitting quality rather than generalization to held-out views. Since current TSDF-based mesh reconstruction relies on depth maps rendered from Gaussians, for models us-

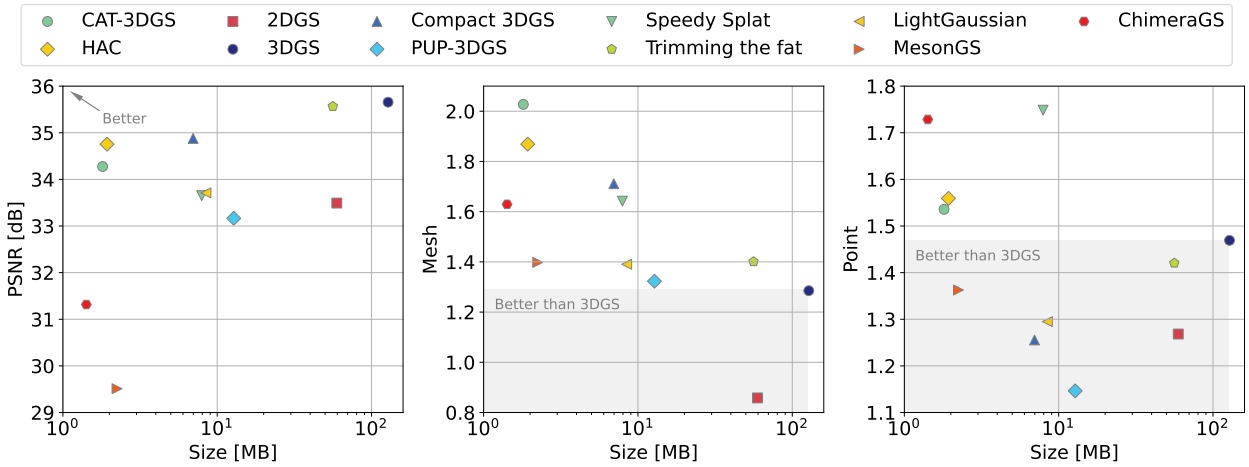


Figure 4. Results of DTU dataset. Here we present the PSNR, along with the Chamfer distance based on mesh and point cloud respectively.

Model	Setting	PSNR [dB]	Size [MB]
PUP-3DGS	Reported	26.67	74.65
	Reproduced	26.84	79.07
	Splatwizard	26.90 (+0.2%)	79.07 ($<$ 0.1%)
Speedy-Splat	Reported	26.94	-
	Reproduced	27.15	74.30
	Splatwizard	27.24 (+0.3%)	75.75 (+2%)

Table 4. Representative reproduction results on Mip-NeRF 360. "Reported" denotes numbers from the original papers, "Reproduced" denotes results obtained from the released codebases under our rerun, and "Splatwizard" denotes the corresponding implementations in our unified framework.

ing standard rasterizers we replace the renderer with depth-enabled variants during reconstruction. Our TSDF fusion follows the 2DGS implementation used in the supplementary code.

Compared to photo-level evaluation, we find that some methods even achieve point cloud-level accuracy that approaches or surpasses the performance of 2DGS. This suggests that compression tasks and geometric reconstruction tasks are not entirely conflicting objectives. In the future, we could explore integrating methods from these two specialized tasks to investigate pathways for enhancing each other's performance. Meanwhile, we should also note that not every compressor can improve geometric reconstruction quality, so the design of specific methods remains particularly important.

5.4. Reproduction Details

To validate that the benchmark faithfully preserves the behavior of the original methods, we compare representative methods across three settings: the results reported in the original paper, the results reproduced from the released official code, and the results obtained after porting the method into Splatwizard. We report both rendering quality and compressed model size, since a benchmark is only useful if it preserves the original rate-distortion trade-off rather than matching a single metric.

Table 4 shows that the reproduced methods remain close after being ported into Splatwizard. For PUP-3DGS, the rendered quality and model size are nearly unchanged. For Speedy-Splat, Splatwizard even yields a small PSNR gain with only a minor increase in compressed size. These results indicate that the framework abstraction itself does not materially distort method behavior.

6. Conclusion and Future Work

In this paper, we introduce Splatwizard, a dedicated benchmark toolkit for 3DGS compression. Splatwizard provides a unified training and evaluation pipeline, flexible module composition, and a comprehensive function library for implementing and comparing compression methods under consistent protocols. Based on this framework, we reproduce multiple representative methods and present ChimeraGS as a case study showing that competitive hybrid pipelines can be rapidly assembled within the same benchmark infrastructure. Our experiments report photometric, computational, and geometric comparisons of current methods. We hope this work will accelerate research in this field by providing fundamental tools and resources. In the future, we will further expand the capabilities of this framework by incorporating a wider range of methods for comparison, while systematically organizing its functional modules to facilitate their use in subsequent work. Additionally, while the current framework is primarily tailored for static GS scene compression methods, we plan to extend its functionality to tasks like dynamic scenes [45, 51], feed-forward paradigm [7] and streaming 3DGS compression [40].

Acknowledgments

This work is supported in part by the National Science and Technology Major Project under grant 2025ZD1601000, National Natural Science Foundation of China under grant 62301189, 62576122, 62571298, 62521006, Guangdong Basic and Applied Basic Research Foundation under grant 2026A1515011139.

References

[1] Muhammad Salman Ali, Maryam Qamar, Sung-Ho Bae, and Enzo Tartaglione. Trimming the fat: Efficient compression of 3d gaussian splats through pruning. *arXiv preprint arXiv:2406.18214*, 2024. 2

[2] M. T. Bagdasarian, P. Knoll, Y. Li, F. Barthel, A. Hilsmann, P. Eisert, and W. Morgenstern. 3dgs.zip: A survey on 3d gaussian splatting compression methods. *Computer Graphics Forum*, page e70078, 2025. https://w-m.github.io/3dgs-compression-survey/. 1, 3

[3] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, 2018. 1

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

[5] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19697–19705, 2023. 2

[6] Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *arXiv preprint arXiv:2406.06521*, 2024. 1

[7] Yihang Chen, Qianyi Wu, Mengyao Li, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Fast feedforward 3d gaussian splatting compression. *arXiv preprint arXiv:2410.08017*, 2024. 8

[8] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Hac: Hash-grid assisted context for 3d gaussian splatting compression. In *European Conference on Computer Vision*, pages 422–438. Springer, 2024. 2, 3, 6

[9] Luca Di Grazia and Michael Pradel. The evolution of type annotations in python: an empirical study. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 209–220, 2022. 3

[10] Lue Fan, Yuxue Yang, Minxing Li, Hongsheng Li, and Zhaoxiang Zhang. Trim 3d gaussian splatting for accurate geometry representation. *arXiv preprint arXiv:2406.07499*, 2024. 4

[11] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, Zhangyang Wang, et al. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *Advances in neural information processing systems*, 37: 140138–140158, 2024. 2, 3, 4, 5

[12] Guofeng Feng, Siyan Chen, Rong Fu, Zimu Liao, Yi Wang, Tao Liu, Boni Hu, Linning Xu, Zhilin Pei, Hengjie Li, et al. Flashgs: Efficient 3d gaussian splatting for large-scale and high-resolution rendering. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26652–26662, 2025. 3, 6

[13] John Flynn, Michael Broxton, Lukas Murmann, Lucy Chai, Matthew DuVall, Clément Godard, Kathryn Heal, Srinivas Kaza, Stephen Lombardi, Xuan Luo, et al. Quark: Real-time, high-resolution, and general neural view synthesis. *ACM Transactions on Graphics (TOG)*, 43(6):1–20, 2024. 2

[14] Yongjie Guan, Xueyu Hou, Nan Wu, Bo Han, and Tao Han. Metastream: Live volumetric content capture, creation, delivery, and rendering in real time. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, New York, NY, USA, 2023. Association for Computing Machinery. 1

[15] Minghao Guo, Bohan Wang, Kaiming He, and Wojciech Matusik. Tetsphere splatting: Representing high-quality geometry with lagrangian volumetric meshes. *arXiv preprint arXiv:2405.20283*, 2024. 1

[16] Xiaoshen Han, Minghuan Liu, Yilun Chen, Junqiu Yu, Xiaoyang Lyu, Yang Tian, Bolun Wang, Weinan Zhang, and Jiangmiao Pang. Re^{3}sim: Generating high-fidelity simulation data via 3d-photorealistic real-to-sim for robotic manipulation. *arXiv preprint arXiv:2502.08645*, 2025. 1

[17] Alex Hanson, Allen Tu, Geng Lin, Vasu Singla, Matthias Zwicker, and Tom Goldstein. Speedy-splat: Fast 3d gaussian splatting with sparse pixels and sparse primitives. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21537–21546, 2025. 2, 4, 5

[18] Alex Hanson, Allen Tu, Vasu Singla, Mayuka Jayawardhana, Matthias Zwicker, and Tom Goldstein. Pup 3d-gs: Principled uncertainty pruning for 3d gaussian splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5949–5958, 2025. 2, 5

[19] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.*, 37(6), 2018. 6

[20] Jan Held, Renaud Vandeghen, Sanghyun Son, Daniel Rebain, Matheus Gadelha, Yi Zhou, Ming C. Lin, Marc Van Droogenbroeck, and Andrea Tagliasacchi. Triangle splatting+: Differentiable rendering with opaque triangles, 2025. 1

[21] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*, pages 1–11, 2024. 1, 3, 4

[22] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014. 6

[23] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time

radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1, 2, 3

[24] 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 (TOG)*, 43(4):1–15, 2024. 4

[25] 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. 6

[26] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21719–21728, 2024. 2, 3, 5

[27] Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. *Advances in Neural Information Processing Systems*, 34:18114–18125, 2021. 1

[28] Ke Li, Reinhard Bacher, Susanne Schmidt, Wim Leemans, and Frank Steinicke. Reality fusion: Robust real-time immersive mobile robot teleoperation with volumetric visual data fusion. *arXiv preprint arXiv:2408.01225*, 2024. 1

[29] Sicheng Li, Chengzhen Wu, Hao Li, Xiang Gao, Yiyi Liao, and Lu Yu. Gscodec studio: A modular framework for gaussian splat compression. *arXiv preprint arXiv:2506.01822*, 2025. 3

[30] Zimu Liao, Jifeng Ding, Siwei Cui, Ruixuan Gong, Boni Hu, Yi Wang, Hengjie Li, XIngcheng Zhang, Hui Wang, and Rong Fu. Tc-gs: A faster gaussian splatting module utilizing tensor cores. *arXiv preprint arXiv:2505.24796*, 2025. 3, 6

[31] Xiang Liu, Bin Chen, Zimo Liu, Yaowei Wang, and Shu-Tao Xia. An exploration with entropy constrained 3d gaussians for 2d video compression. In *The Thirteenth International Conference on Learning Representations*, 2025. 2

[32] Xiang Liu, Jiahong Chen, Bin Chen, Zimo Liu, Baoyi An, Shu-Tao Xia, and Zhi Wang. An efficient implicit neural representation image codec based on mixed autoregressive model for low-complexity decoding. *IEEE Transactions on Multimedia*, 2025. 1

[33] 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*, pages 1–11, 2024. 3

[34] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 6

[35] 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. *Communications of the ACM*, 65(1):99–106, 2021. 2

[36] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10349–10358, 2024. 2

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 3

[38] Shiyu Qin, Jinpeng Wang, Yimin Zhou, Bin Chen, Tianci Luo, Baoyi An, Tao Dai, Shutao Xia, and Yaowei Wang. Mambavc: Learned visual compression with selective state spaces. *arXiv preprint arXiv:2405.15413*, 2024. 1

[39] Shiyu Qin, Jinpeng Wang, Yimin Zhou, Bin Chen, Tianci Luo, Baoyi An, Tao Dai, Shu-Tao Xia, and Yaowei Wang. Cassic: Towards content-adaptive state-space models for learned image compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15727–15736, 2025. 1

[40] Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 3dgstream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20675–20685, 2024. 8

[41] Henan Wang, Hanxin Zhu, Tianyu He, Runsen Feng, Jiajun Deng, Jiang Bian, and Zhibo Chen. End-to-end rate-distortion optimized 3d gaussian representation. In *European Conference on Computer Vision*, pages 76–92. Springer, 2024. 5

[42] Penghao Wang, Zhirui Zhang, Liao Wang, Kaixin Yao, Siyuan Xie, Jingyi Yu, Minye Wu, and Lan Xu. V^3: Viewing volumetric videos on mobiles via streamable 2d dynamic gaussians. *CoRR*, abs/2409.13648, 2024. 1

[43] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4690–4699, 2021. 2

[44] 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

[45] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20310–20320, 2024. 8

[46] Yichong Xia, Yimin Zhou, Jinpeng Wang, Baoyi An, Haoqian Wang, Yaowei Wang, and Bin Chen. Diffpc: Diffusion-based high perceptual fidelity image compression with semantic refinement. In *The Thirteenth International Conference on Learning Representations*, 2025. 1

[47] Yichong Xia, Yimin Zhou, Jinpeng Wang, and Bin Chen. Towards efficient low-rate image compression with frequency-aware diffusion prior refinement. *arXiv preprint arXiv:2601.10373*, 2026. 1

[48] 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 *The European Conference on Computer Vision (ECCV)*, 2022. 6

[49] Shuzhao Xie, Weixiang Zhang, Chen Tang, Yunpeng Bai, Rongwei Lu, Shijia Ge, and Zhi Wang. Mesongs: Post-training compression of 3d gaussians via efficient attribute transformation. In *European Conference on Computer Vision*. Springer, 2024. 2

[50] Shuzhao Xie, Jiahang Liu, Weixiang Zhang, Shijia Ge, Sicheng Pan, Chen Tang, Yunpeng Bai, Cong Zhang, Xiaoyi Fan, and Zhi Wang. Sizegs: Size-aware compression of 3d gaussian splatting via mixed integer programming. In *ACM MM*, 2025. 2

[51] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20331–20341, 2024. 8

[52] Yuxuan Yao, Zixuan Zeng, Chun Gu, Xiatian Zhu, and Li Zhang. Reflective gaussian splatting. *arXiv preprint arXiv:2412.19282*, 2024. 2

[53] Chongjie Ye, Yinyu Nie, Jiahao Chang, Yuantao Chen, Yihao Zhi, and Xiaoguang Han. Gaustudio: A modular framework for 3d gaussian splatting and beyond. *arXiv preprint arXiv:2403.19632*, 2024. 1, 3

[54] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, et al. gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025. 3

[55] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19447–19456, 2024. 2

[56] Yu-Ting Zhan, Cheng-Yuan Ho, Hebi Yang, Yi-Hsin Chen, Jui Chiu Chiang, Yu-Lun Liu, and Wen-Hsiao Peng. Cat-3dgs: A context-adaptive triplane approach to rate-distortion-optimized 3dgs compression. *arXiv preprint arXiv:2503.00357*, 2025. 2, 3

[57] Fengdi Zhang, Hongkun Cao, and Ruqi Huang. Consistent quantity-quality control across scenes for deployment-aware gaussian splatting. *arXiv preprint arXiv:2505.10473*, 2025. 2

[58] Jiakai Zhang, Liao Wang, Xinhang Liu, Fuqiang Zhao, Minzhang Li, Haizhao Dai, Boyuan Zhang, Wei Yang, Lan Xu, and Jingyi Yu. Neuvv: Neural volumetric videos with immersive rendering and editing. *CoRR*, abs/2202.06088, 2022. 1

[59] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 4

[60] Xinjie Zhang, Xingtong Ge, Tongda Xu, Dailan He, Yan Wang, Hongwei Qin, Guo Lu, Jing Geng, and Jun Zhang. Gaussianimage: 1000 fps image representation and compression by 2d gaussian splatting. *arXiv preprint arXiv:2403.08551*, 2024. 2

[61] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 4