3DGS.zip: A survey on 3D Gaussian Splatting Compression Methods

Milena T. Bagdasarian[1], Paul Knoll[1], Yi-Hsin Li[3], Florian Barthel[1,2], Anna Hilsmann[1], Peter Eisert[1,2], and Wieland Morgenstern[1]

[1]Fraunhofer Heinrich Hertz, HHI
[2]Humboldt University of Berlin
[3]Technical University Berlin

Abstract

We present a work-in-progress survey on 3D Gaussian Splatting [**?**] compression methods, focusing on their statistical performance across various benchmarks. This survey aims to facilitate comparability by summarizing key statistics of different compression approaches in a tabulated format. The datasets evaluated include TanksAndTemples [14], MipNeRF360 [1], DeepBlending [10], and SyntheticNeRF [21]. For each method, we report the Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), Learned Perceptual Image Patch Similarity (LPIPS), and the resultant size in megabytes (MB), as provided by the respective authors. This is an ongoing, open project, and we invite contributions from the research community as GitHub issues or pull requests. Please visit `http://w-m.github.io/3dgs-compression-survey/` for more information and a sortable version of the table.

1 Scope of this survey

In this survey, we focus on compression methods for 3D Gaussian Splatting (3DGS), aiming to optimize memory usage while preserving visual quality and real-time rendering speed. We provide a comprehensive comparison of various compression techniques, with quantitative results for the most commenly used datasets summarized in a tabulated format. Our goal is to ensure transparency and reproducibility of the included approaches. Additionally, we offer a brief explanation of each pipeline and discuss main compression approaches. Rather than covering all existing 3DGS methods, our focus is specifically on their compression techniques; for a broader overview of 3DGS methods and applications, we refer readers to [7, 32]. While we include many common approaches shared between neural radiance field (NeRF) [20] compression and 3DGS compression, we direct readers to [3, 16] for NeRF-specific compression methods.

Survey Table

Method	Rank	TanksAndTemples				MipNeRF360				DeepBlending				SyntheticNeRF			
		PSNR↑	SSIM↑	LPIPS↓	Size MB↓	PSNR↑	SSIM↑	LPIPS↓	Size MB↓	PSNR↑	SSIM↑	LPIPS↓	Size MB↓	PSNR↑	SSIM↑	LPIPS↓	Size MB↓
HAC-highrate	3.7	24.40	0.853	0.177	11.2	27.77	0.811	0.230	21.9	30.34	0.906	0.258	6.3	33.71	0.968	0.034	1.9
HAC-lowrate	4.1	24.04	0.846	0.187	8.1	27.53	0.807	0.238	15.3	29.98	0.902	0.269	4.3	33.24	0.967	0.037	1.2
gsplat-1.00M	4.3	24.03	0.857	0.163	15.4	27.29	0.811	0.229	15.3								
IGS low	5.3	23.70	0.836	0.227	8.4	27.34	0.811	0.255	12.8	30.63	0.904	0.293	6.3	33.36	0.971	0.036	1.9
IGS high	5.7	24.05	0.849	0.211	12.5	27.62	0.820	0.245	25.8	32.33	0.924	0.253	7.7	34.18	0.975	0.032	2.7
Navaneet et al. 32K	7.2	23.44	0.838	0.198	13.0	27.12	0.806	0.240	19.0	29.90	0.907	0.251	13.0				
Navaneet et al. 16K	7.6	23.39	0.836	0.200	12.0	27.03	0.804	0.243	18.0	29.90	0.906	0.252	12.0				
RDO-Gaussian	8.2	23.34	0.835	0.195	11.5	27.05	0.802	0.239	22.4	29.63	0.902	0.252	17.2	33.12	0.967	0.035	2.2
Reduced3DGS	8.6	23.57	0.840	0.188	14.0	27.10	0.809	0.226	29.0	29.63	0.902	0.249	18.0				
Morgenstern et al. w/o SH	9.2	23.15	0.828	0.198	8.9	26.56	0.791	0.241	15.9	29.12	0.892	0.270	5.4	31.37	0.959	0.043	1.9
MesonGS c3	10.5	23.29	0.835	0.197	16.6	26.99	0.797	0.246	24.7	29.48	0.903	0.252	27.7	32.96	0.968	0.033	3.3
Compressed3D	11.1	23.32	0.832	0.194	17.3	26.98	0.801	0.238	28.8	29.38	0.898	0.253	25.3	32.94	0.967	0.033	3.7
MesonGS c1	11.2	23.31	0.835	0.196	17.6	26.99	0.796	0.247	27.2	29.50	0.903	0.251	29.6	32.94	0.968	0.033	3.7
Morgenstern et al.	11.3	23.56	0.837	0.186	21.7	27.08	0.799	0.230	38.4	29.26	0.894	0.268	16.9	33.23	0.966	0.034	3.9
Compact3DGS+PP	11.9	23.32	0.831	0.202	20.9	27.03	0.797	0.247	29.1	29.73	0.900	0.258	23.8	32.88	0.968	0.034	2.7
EAGLES	12.6	23.37	0.84	0.20	29.0	27.23	0.81	0.24	54.0	29.86	0.91	0.25	52.0				
Scaffold-GS	12.9	23.96	0.853	0.177	87.0	27.50	0.806	0.252	253.9	30.21	0.906	0.254	66.0				
Compact3DGS	13.7	23.32	0.831	0.201	39.4	27.08	0.798	0.247	48.8	29.79	0.901	0.258	43.2	33.33	0.968	0.034	5.5
LightGaussian	14.4	23.11	0.817	0.231	22.0	27.28	0.805	0.243	42.0					32.72	0.965	0.037	7.8
EAGLES-Small	14.9	23.10	0.82	0.22	19.0	26.94	0.80	0.25	47.0	29.92	0.90	0.25	33.0				

Note: The best methods in each category are highlighted (fist , second , third). The ranks represent the average rankings of the methods across all available datasets. The quality metrics PSNR, SSIM, and LPIPS are equally weighted with the model size, meaning they each contribute one-sixth to the ranks, while the size contributes half.

2 Datasets and Evaluation Statistics

2.1 Datasets

Performance and quality assessment of 3D Gaussian Splatting algorithms is typically performed on multiple datasets. These datasets provide 3D scenes or objects with various properties, such as varying levels of detail, lighting conditions, and complexities, which allow for comprehensive evaluation of the algorithms.
In our survey we include Tanks and Temples [14], MipNerf360 [1], Deep Blending [10] as real-world datasets, and Synthetic NeRF [21] as a synthetic dataset. From Tanks and Temples we include "truck" and "train" two unbounded outdoor scenes wich have a centered view point. The MipNerf360 dataset also has a centered view point but includes in- and outdoor scenes. The following scenes are included: "bicycle", "bonsai", "counter", "flowers", "garden", "kitchen", "room", "stump", "treehill". From the Deep Blending dataset we include "Dr Johnson" and "Palyroom" two indoor scenes with a viewpoint directed outward. The synthetic scenes: chair, drums, ficus, hotdog, lego, material, mic, ship stem from the SyntheticNeRF dataset.

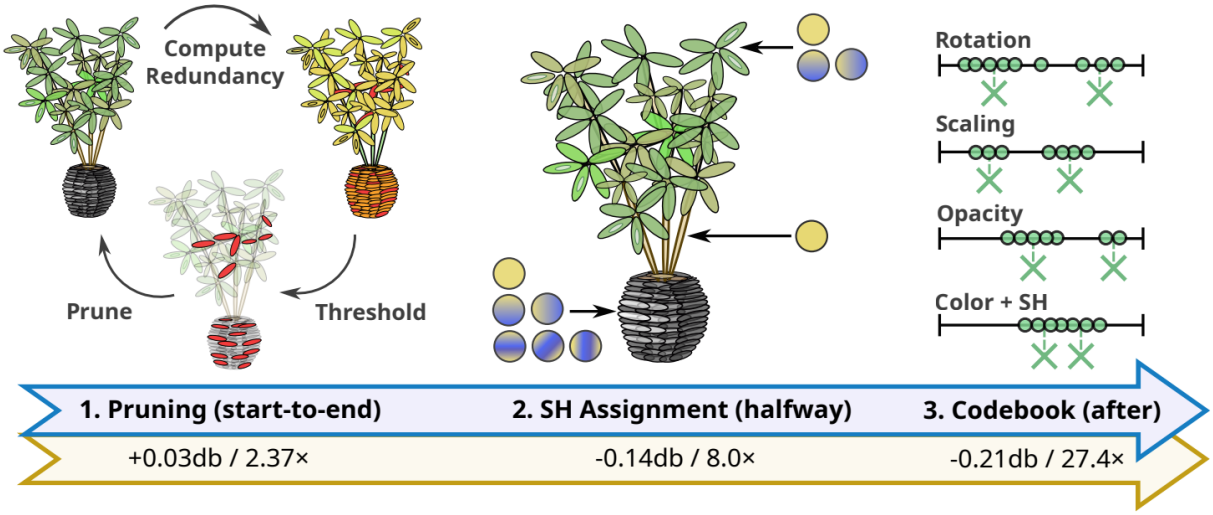
2.2 Evaluation Statistics

3 Description of Included Compression Approaches

In the following sections, we present a comprehensive overview of state-of-the-art 3D Gaussian Splatting (3DGS) compression methods. Although each method shares the common goal of minimizing memory usage while preserving rendering quality and speed, their strategies differ significantly. The summaries aim to distill the key concepts behind each approach, enabling better comparison and serving as a valuable reference for future research in the field of 3DGS compression.

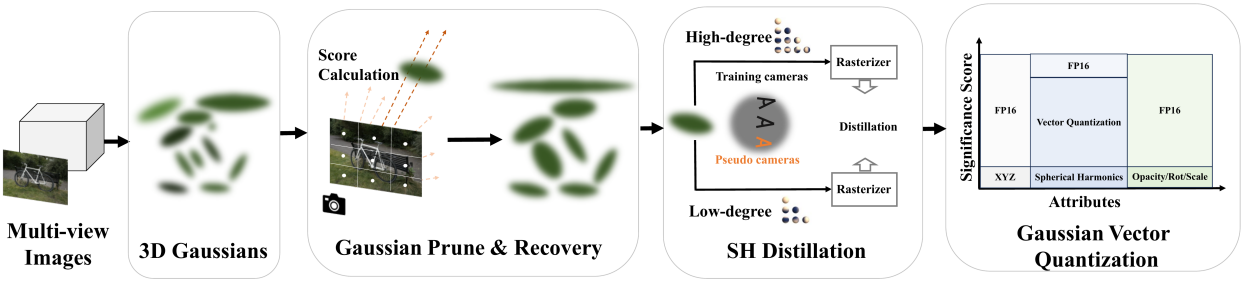
3.1 Reducing the Memory Footprint of 3D Gaussian Splatting

This approach addresses three main issues contributing to large storage sizes in 3D Gaussian Splatting (3DGS). To reduce the number of 3D Gaussian primitives, the authors introduce a scale- and resolution-aware redundant primitive removal method. This extends opacity-based pruning by incorporating a redundancy score to identify regions with many low-impact primitives. To mitigate storage size due to spherical harmonic coefficients, they propose adaptive adjustment of spherical harmonic (SH) bands. This involves evaluating color consistency across views and reducing higher-order SH bands when view-dependent effects are minimal. Additionally, recognizing the limited need for high dynamic range and precision for most primitive attributes, they develop a codebook using K-means clustering and apply 16-bit half-float quantization to the remaining uncompressed floating point values. [25]



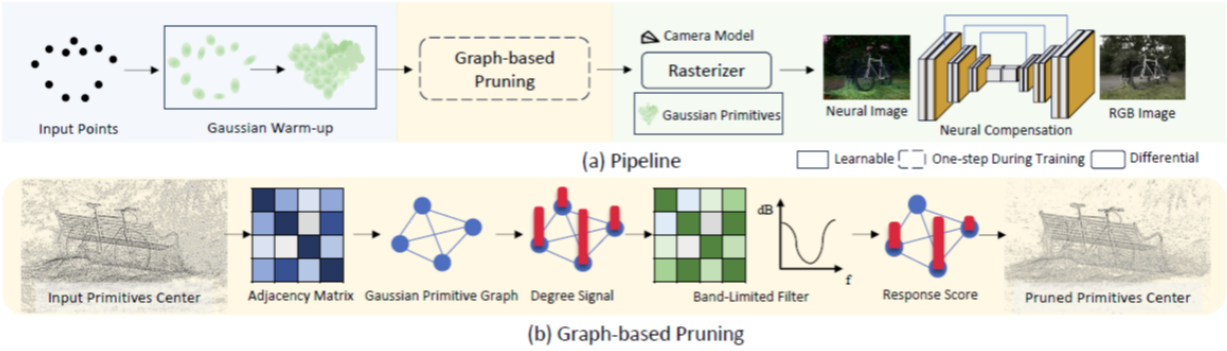
3.2 LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS

LightGaussian aims to transform 3D Gaussians to a more efficient and compact form, avoiding the scalablity issues that arrises from the large number of SfM (Structure from Motion) points for unbounded scenes. Inspired by Network Pruning, the method identifies Gaussians that minimally contribute to scene reconstruction and employs a pruning and recovery process, thereby efficiently reducing redundancy in Gaussian counts while maintaining visual effects. Additionally, LightGaussian utilizes knowledge distillation and pseudo-view augmentation to transfer spherical harmonics efficients to a lower degree. Furthermore, the authors propose a Gaussian Vector Quantization based on the global significance of Gaussians to quantize all redundant attributes, achieving lower bitwidth representations with minimal accuracy losses. [5]



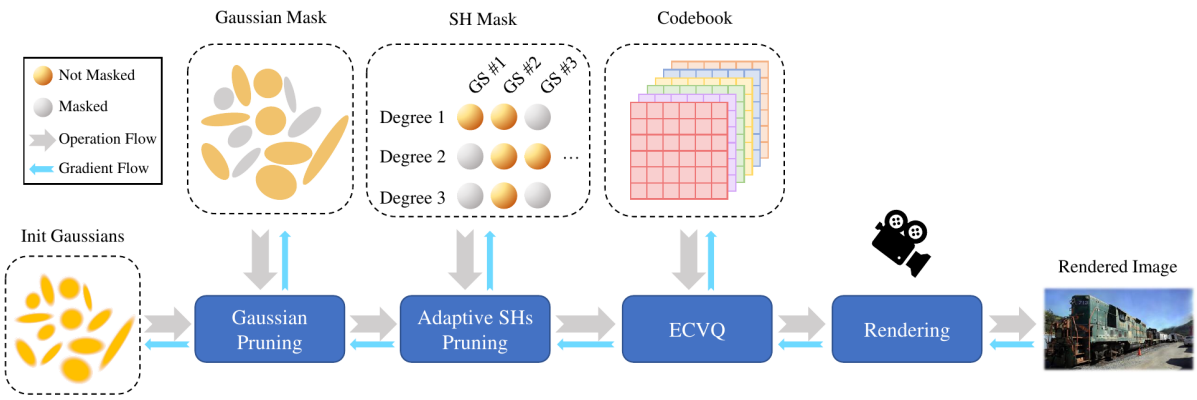
3.3 SUNDAE: Spectrally Pruned Gaussian Fields with Neural Compensation

This method combines graph-based pruning of Gaussian primitives with a convolutional neural network to retain high-frequency details. First, it "warms up" the 3D Gaussian field before applying a graph-based pruning strategy. The pruning step uses a graph built on the spatial relationships between the Gaussians and applies a band-limited graph filter to selectively down-sample, preserving important features. A convolutional neural network is then used to compensate for any losses caused by the pruning, ensuring that both high-frequency details and general low-frequency features are retained. [35]



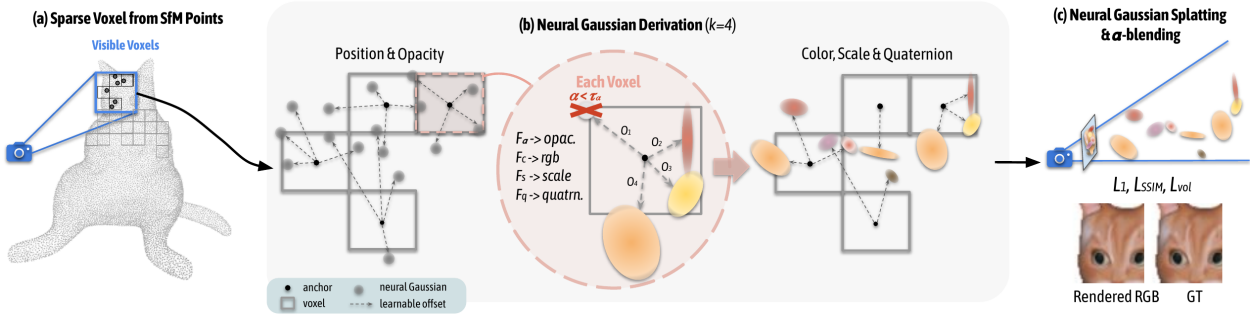
3.4 End-to-End Rate-Distortion Optimized 3D Gaussian Representation

This paper introduces RDO-Gaussian, an end-to-end Rate-Distortion Optimized 3D Gaussian representation. The authors achieve flexible, continuous rate control by formulating 3D Gaussian representation learning as a joint optimization of rate and distortion. Rate-distortion optimization is realized through dynamic pruning and entropy-constrained vector quantization (ECVQ). Gaussian pruning involves learning a mask to eliminate redundant Gaussians and adaptive SHs pruning assigns varying SH degrees to each Gaussian based on material and illumination needs. The covariance and color attributes are discretized through ECVQ, which performs vector quantization. [30]



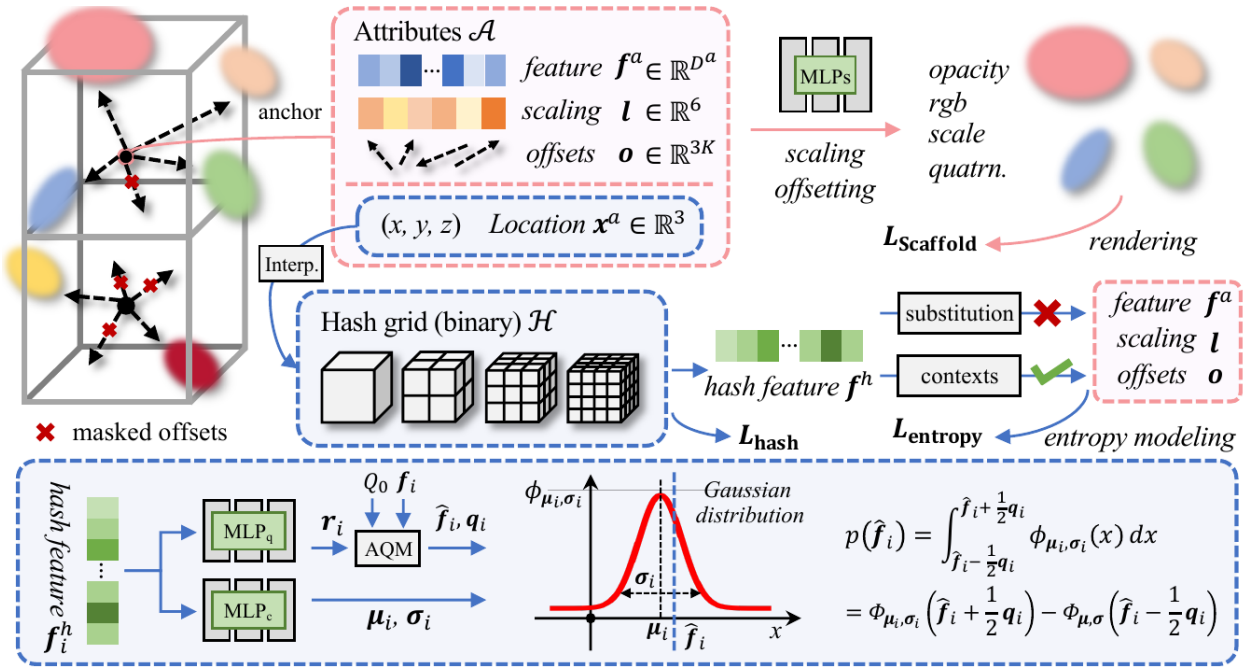
3.5 Scaffold-GS: Structured 3D Gaussians for View-Adaptive Rendering

Scaffold-GS introduces anchor points that leverage scene structure to guide the distribution of local 3D Gaussians. Attributes like opacity, color, rotation, and scale are dynamically predicted for Gaussians linked to each anchor within the viewing frustum, enabling adaptation to different viewing directions and distances. Initial anchor points are derived by voxelizing the sparse, irregular point cloud from Structure from Motion (SfM), forming a regular grid. To refine and grow the anchors, Gaussians are spatially quantized using voxels, with new anchors created at the centers of significant voxels, identified by their average gradient over N training steps. Random elimination and opacity-based pruning regulate anchor growth and refinement. [19]



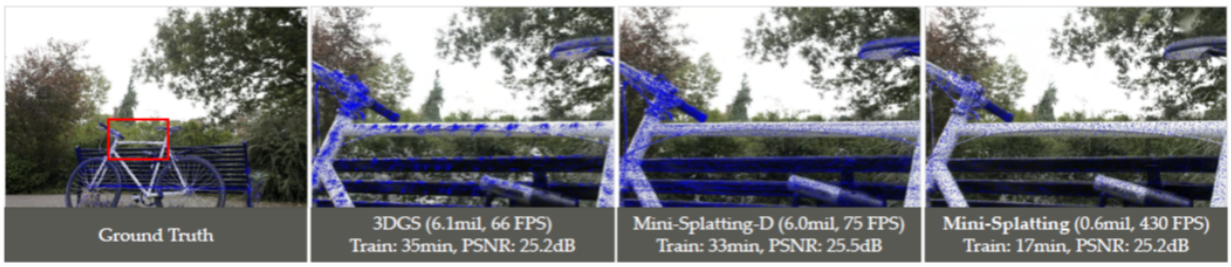
3.6 HAC: Hash-grid Assisted Context for 3D Gaussian Splatting Compression

The paper proposes a Hash-grid Assisted Context (HAC) framework for compressing 3D Gaussian Splatting (3DGS) models by leveraging the mutual information between attributes of unorganized 3D Gaussians (anchors) and hash grid features. Using Scaffold-GS as a base model, HAC queries the hash grid by anchor location to predict anchor attribute distributions for efficient entropy coding. The framework introduces an Adaptive Quantization Module (AQM) to dynamically adjust quantization step sizes. Furthermore, this method employs adaptive offset masking with learnable masks to eliminate invalid Gaussians and anchors, by leveraging the pruning strategy introduced by Compact3DGS and additionally removing anchors if all the attached offsets are pruned. [2]



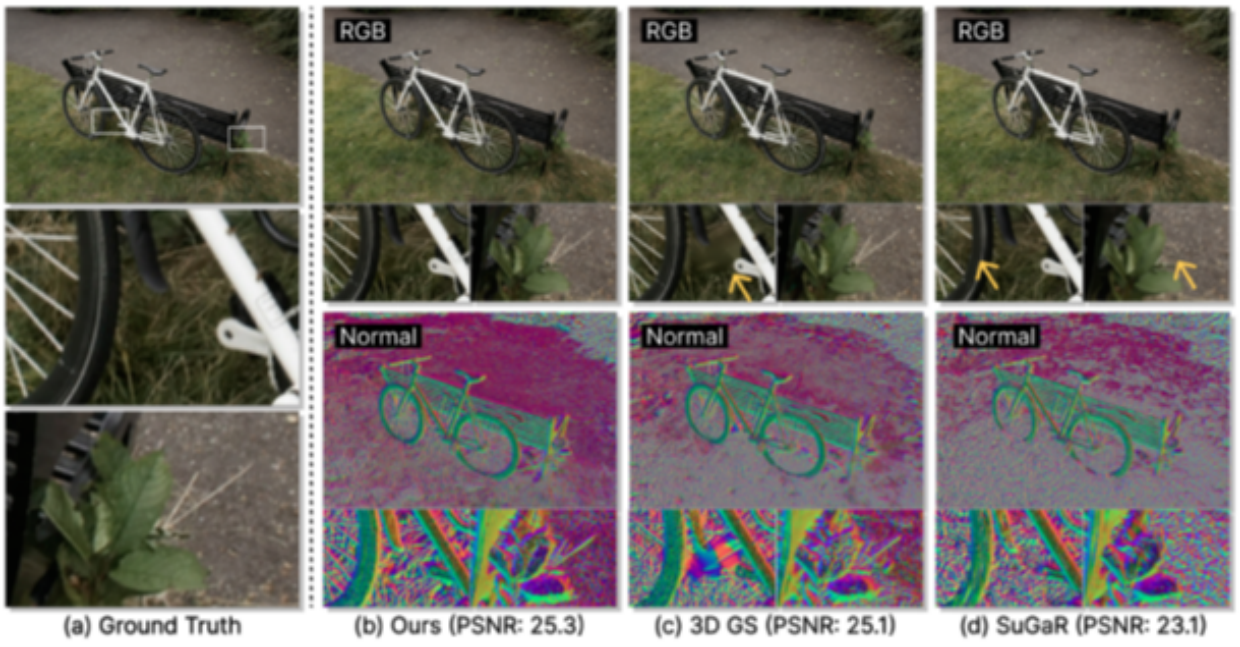
3.7 Mini-Splatting: Representing Scenes with a Constrained Number of Gaussians

Mini-Splatting enhances Gaussian distribution through Blur Split, which refines Gaussians in blurred regions, and Depth Reinitialization, which repositions Gaussians based on newly generated depth points, calculated from the mid-point of ray intersections with Gaussian ellipsoids, thus avoiding artifacts from alpha blending. For simplification, Intersection Preserving retains Gaussians with the greatest visual impact, while Sampling maintains geometric integrity and rendering quality, reducing complexity. [6]



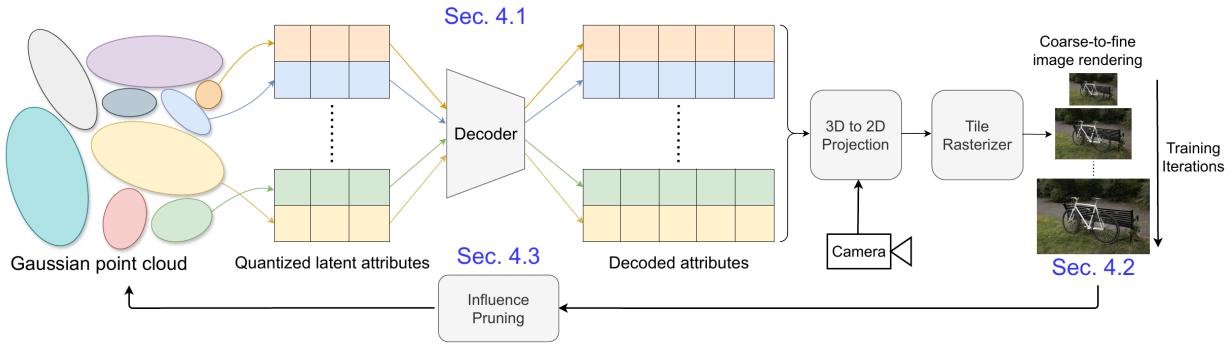
3.8 AtomGS: Atomizing Gaussian Splatting for High-Fidelity Radiance Field

This method prioritizes fine details through Atom Gaussians, which are isotropic and uniformly sized to align closely with the scene's geometry, while large Gaussians are merged to cover smooth surfaces. In addition, Geometry-Guided Optimization uses an Edge-Aware Normal Loss and multi-scale SSIM to maintain geometric accuracy. The Edge-Aware Normal Loss is calculated as the product of the normal map, derived from the pre-optimized 3DGS, and the edge map, which is derived from the gradient magnitude of the ground truth RGB image. [18]



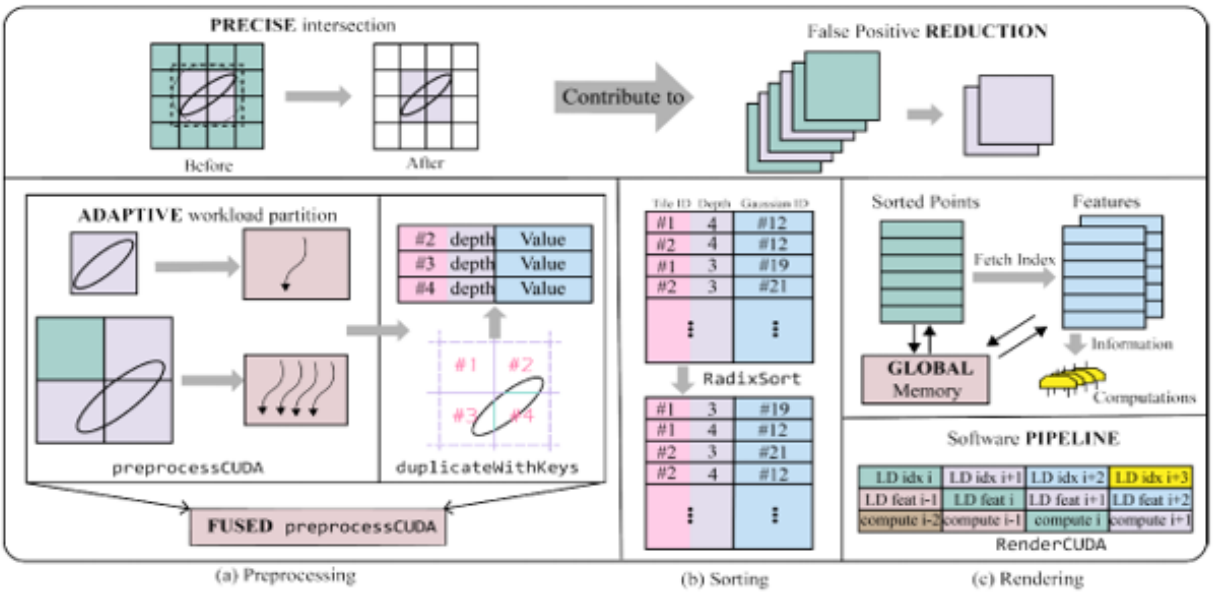
3.9 EAGLES: Efficient Accelerated 3D Gaussians with Lightweight EncodingS

The authors of this approach observed that in 3DGS, the color and rotation attributes account for over 80% of memory usage; thus, they propose compressing these attributes via a latent quantization framework. Additionally, they quantize the opacity coefficients of the Gaussians, improving optimization and resulting in fewer floaters or visual artifacts in novel view reconstructions. To reduce the number of redundant Gaussians resulting from frequent densification (via cloning and splitting), the approach employs a pruning stage to identify and remove Gaussians with minimal influence on the full reconstruction. For this, an influence metric is introduced, which considers both opacity and transmittance. [9]



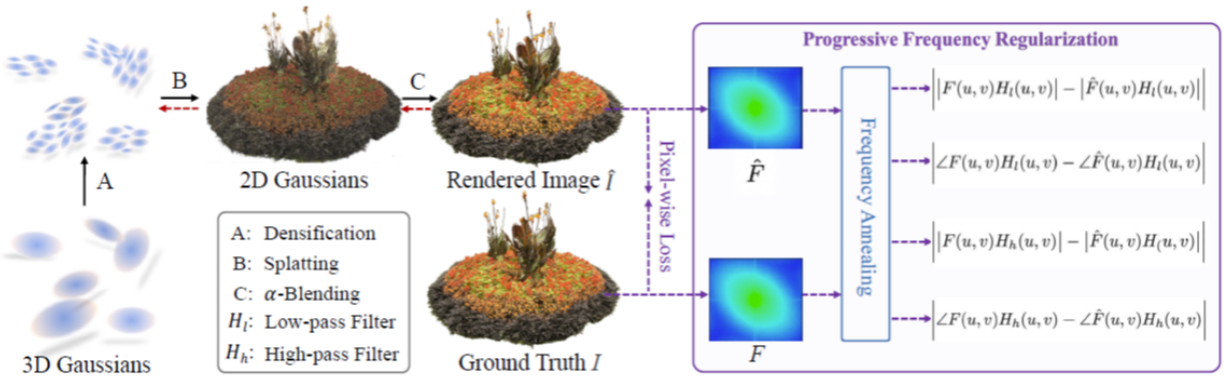
3.10 FlashGS: Efficient 3D Gaussian Splatting for Large-scale and High-resolution Rendering

The method reduces computational overhead by addressing inefficiencies in direct intersection calculations for elongated ellipses. It employs a two-stage filtering process: (1) a tightly tangent bounding box is used to set a coarse range, and (2) tiles intersecting the bounding box are checked for ellipse intersections. Additionally, FlashGS reduces global memory queries by employing a software pipelining approach that overlaps instruction dispatch, optimizing data retrieval and rendering efficiency. [8]



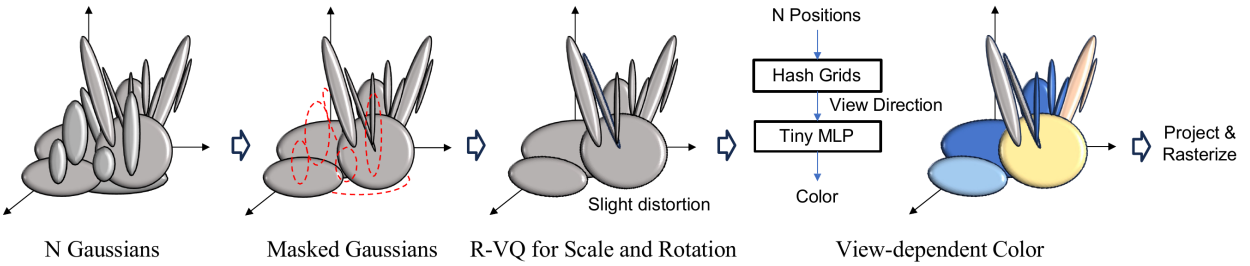
3.11 FreGS: 3D Gaussian Splatting with Progressive Frequency Regularization

This approach is designed to address over-reconstruction from a frequency-based perspective. It minimizes the discrepancy between the frequency spectra of rendered images and the corresponding ground truth. This method explicitly regularizes amplitude and phase differences within Fourier space to achieve more accurate results. [36]



3.12 Compact 3D Gaussian Representation for Radiance Field

This approach introduces a Gaussian volume mask to prune non-essential Gaussians and a compact attribute representation for both view-dependent color and geometric attributes. The volume-based masking strategy combines opacity and scale to selectively remove redundant Gaussians. For color attribute compression, spatial redundancy is exploited by incorporating a grid-based (Instant-NGP) neural field, allowing efficient representation of view-dependent colors without storing attributes per Gaussian. Given the limited variation in scale and rotation, geometric attribute compression employs a compact codebook-based representation to identify and reuse similar geometries across the scene. Additionally, the authors propose quantization and entropy-coding as post-processing steps for further compression. [15]



3.13 gsplat

This approach leverages 3D Gaussian Splatting as Markov Chain Monte Carlo (3DGS-MCMC), interpreting the training process of positioning and optimizing Gaussians as a sampling procedure rather than minimizing a predefined loss function.

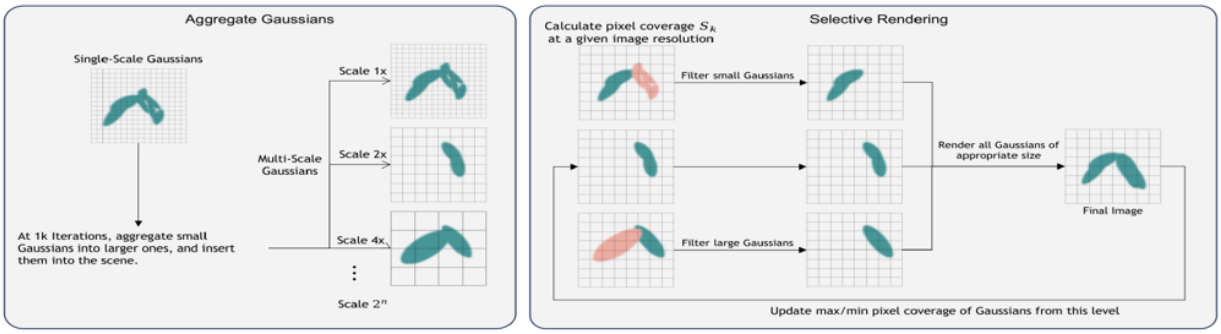
Additionally, it incorporates compression techniques derived from the Morgenstern et al. paper, which organizes the parameters of 3DGS in a 2D grid, capitalizing on perceptual redundancies found in natural scenes, thereby significantly reducing storage requirements.

Further compression is achieved by applying methods from Making Gaussian Splats more smaller, which reduces the size of Gaussian splats by clustering spherical harmonics into discrete elements and storing them as FP16 values.

This technique is implemented in gsplat, an open-source library designed for CUDA-accelerated differentiable rasterization of 3D Gaussians, equipped with Python bindings. [11]

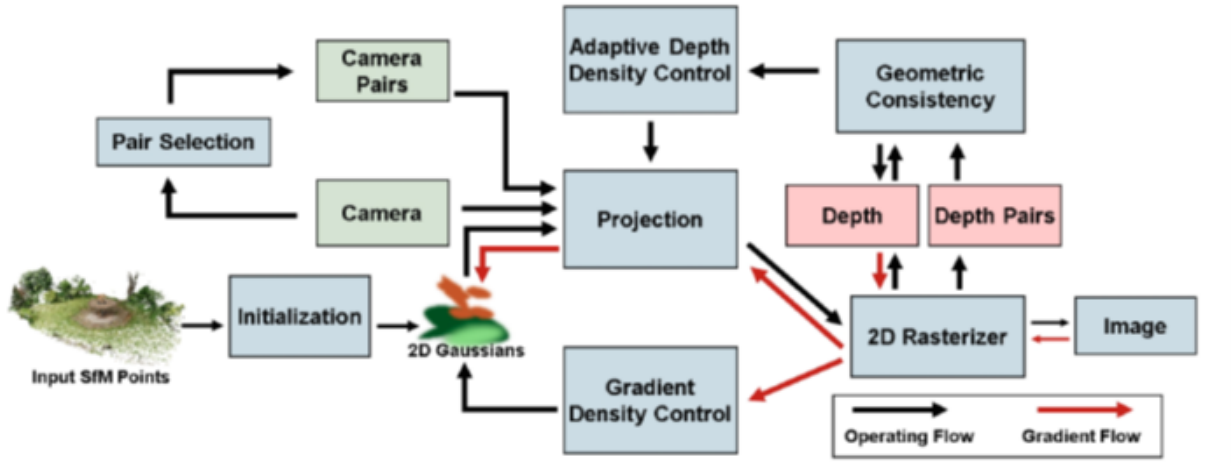
3.14 Multi-Scale 3D Gaussian Splatting for Anti-Aliased Rendering

This method aggregates small Gaussians below a size threshold during early training, enlarging and inserting them into the scene at various resolution scales. These multi-scale Gaussians are then selected for rendering based on their "pixel coverage" at the current resolution. During rendering, only Gaussians with pixel coverage between adaptively selected minimum and maximum are selected. These pixel-coverage values are dynamically updated based on the aggregation results from the first stage. [34]



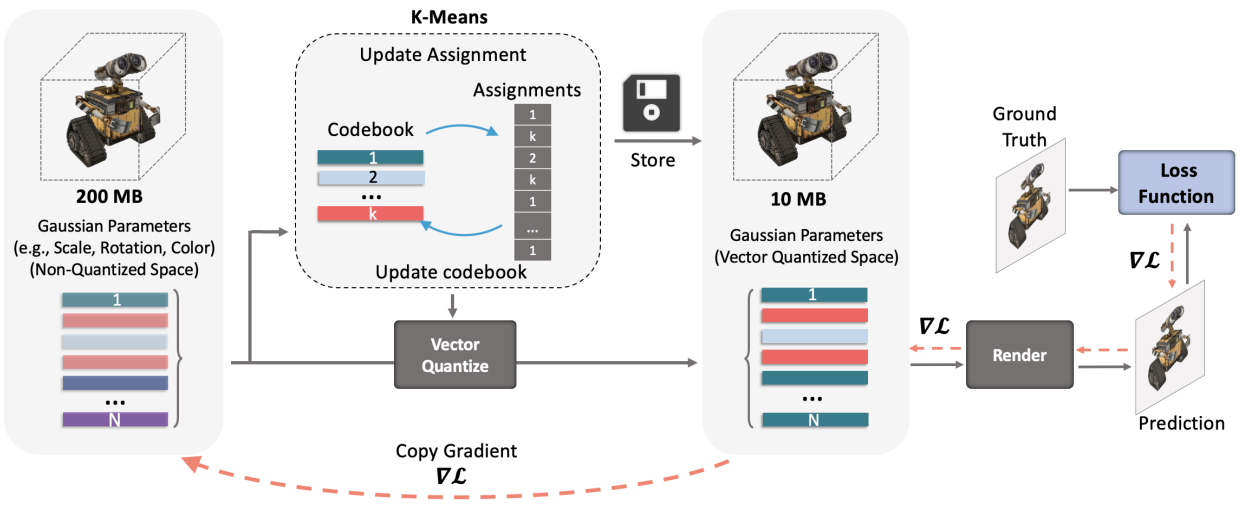
3.15 MVG-Splatting: Multi-View Guided Gaussian Splatting with Adaptive Quantile-Based Geometric Consistency Densification

This method introduces adaptive depth density control to enhance 2D GS. 2D GS renders depth maps. In this method, the render depth maps are dynamically divided into near, mid, and far regions based on Kernel Density Estimation (KDE) and Fast Fourier Transform (FFT), focusing densification in under-reconstructed near and far areas. Geometric consistency checks and depth projection ensure adaptive densification without over-reconstruction, particularly in these critical regions. [17]



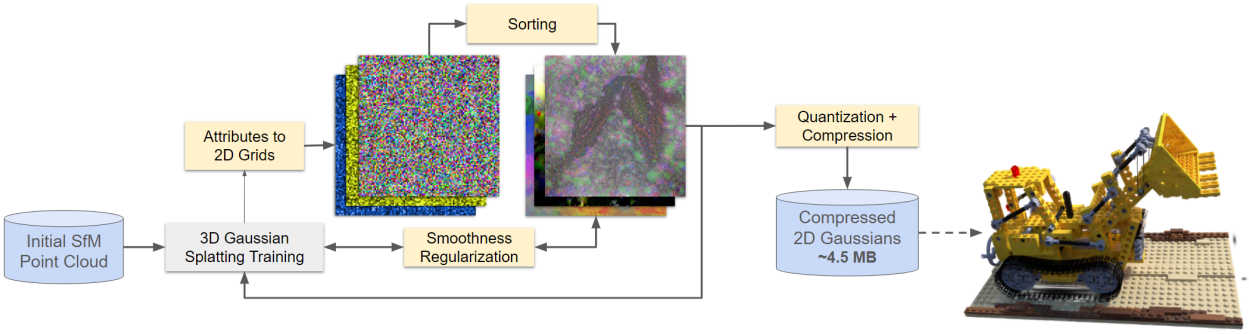
3.16 Compact3D: Compressing Gaussian Splat Radiance Field Models with Vector Quantization

This approach introduces a vector quantization method based on the K-means algorithm to quantize the Gaussian parameters in 3D Gaussian splatting, as many Gaussians may share similar parameters. Only a small codebook is stored along with the index of the code for each Gaussian, resulting in a large reduction in the storage of the learned radiance fields and a reduction of the memory footprint at rendering time. Additionally, the indices are further compressed by sorting the Gaussians based on one of the quantized parameters and storing the indices using a method similar to Run-Length-Encoding (RLE). To reduce the number of Gaussians, this method applies a regularizer to encourage zero opacity, before pruning Gaussians with opacity smaller than a threshold. [23]



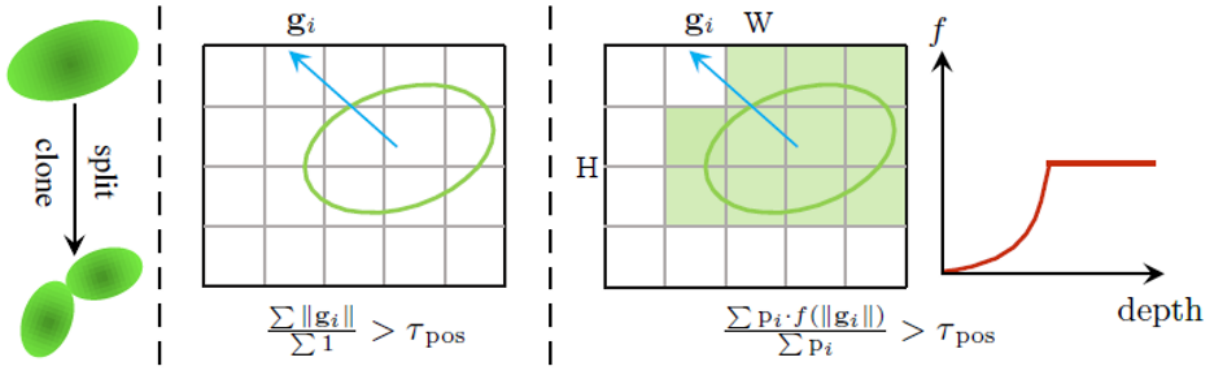
3.17 Compact 3D Scene Representation via Self-Organizing Gaussian Grids

Compressing 3D data is challenging, but many effective solutions exist for compressing 2D data (such as images). The authors propose a new method to organize 3DGS parameters into a 2D grid, drastically reducing storage requirements without compromising visual quality. This organization exploits perceptual redundancies in natural scenes. They introduce a highly parallel sorting algorithm, PLAS, which arranges Gaussian parameters into a 2D grid, maintaining local neighborhood structure and ensuring smoothness. This solution is particularly innovative because no existing method efficiently handles a 2D grid with millions of points. During training, a smoothness loss is applied to enforce local smoothness in the 2D grid, enhancing the compressibility of the data. The key insight is that smoothness needs to be enforced during training to enable efficient compression. [22]



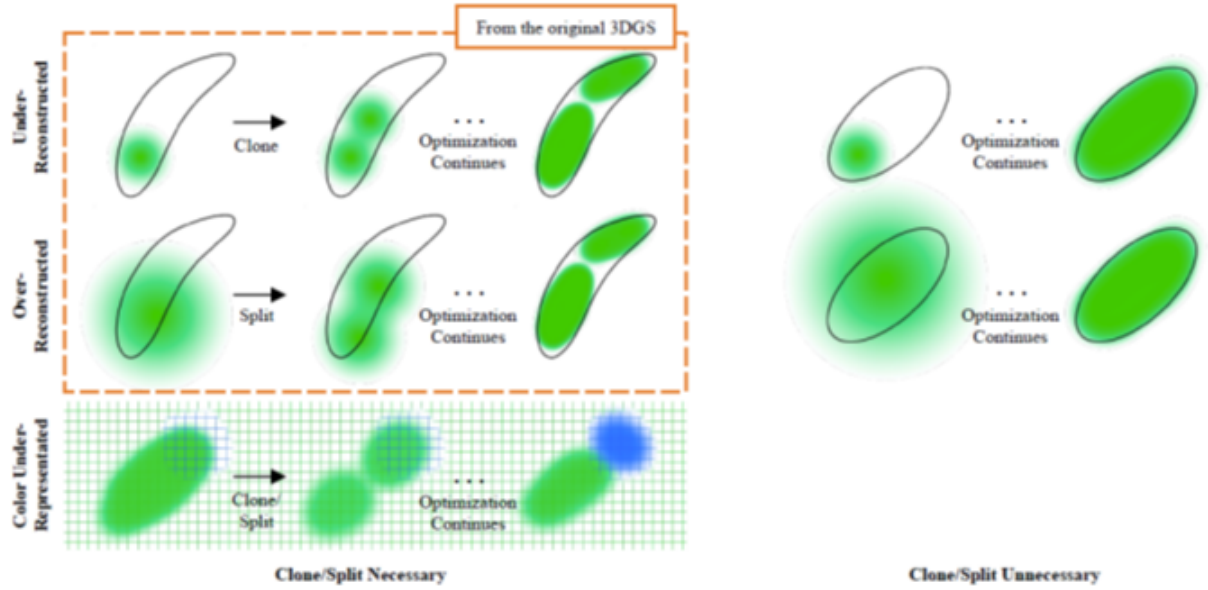
3.18 Pixel-GS: Density Control with Pixel-aware Gradient for 3D Gaussian Splatting

The Pixel-GS method introduces a pixel-aware gradient to address under-reconstruction and needle-like artifacts in 3DGS. This gradient guides densification when the averaged scale gradient across multiple views exceeds a threshold, unlike the original method that only considers a single view. Additionally, Pixel-GS proposes a depth-dependent gradient scaling strategy to balance the influence of Gaussians on affected areas, as near-view Gaussians often receive an excessive number of gradients due to affecting too many pixels. [37]



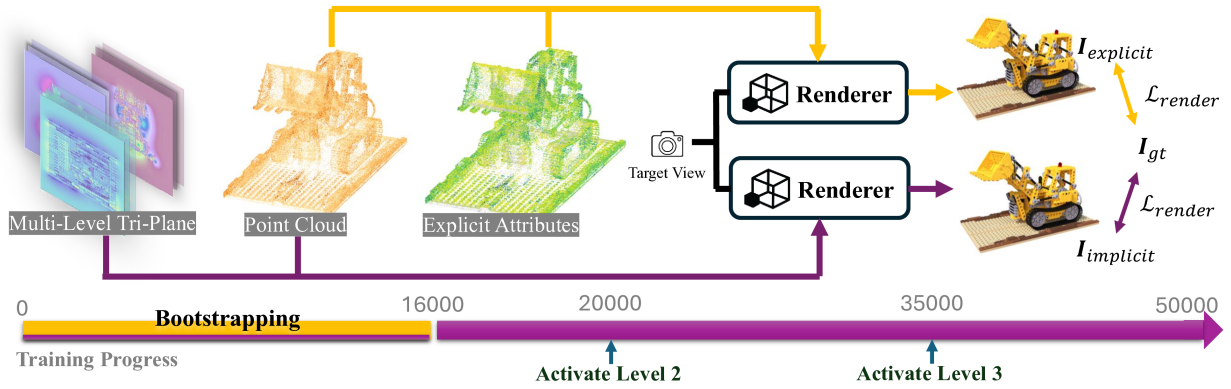
3.19 Color-cued Efficient Densification Method for 3D Gaussian Splatting

This method introduces a simple yet effective modification to the densification process in the original 3D Gaussian Splatting (3DGS). It leverages the view-independent (0th) spherical harmonics (SH) coefficient gradient to better assess color cues for densification, while using the 2D position gradient more coarsely to refine areas where structure-from-motion (SfM) struggles to capture fine structures. [13]



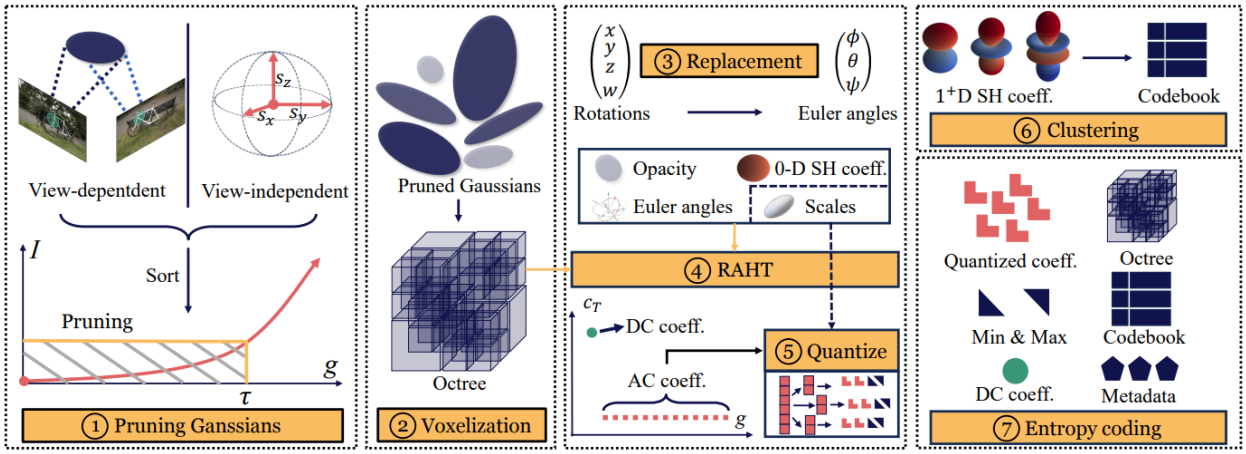
3.20 Implicit Gaussian Splatting with Efficient Multi-Level Tri-Plane Representation

This method introduces a hybrid representation for splatting-based radiance fields, where Gaussian primitives are separated into explicit point cloud and implicit attribute features. The attribute features are encoded using a multi-resolution multi-level tri-plane architecture integrated with a residual-based rendering pipeline. It employs a level-based progressive training scheme for joint optimization of point clouds and tri-planes, starting with coarse attributes and refining them with higher-level details. Spatial regularization and a bootstrapping scheme are applied to enhance the consistency and stability of the Gaussian attributes during training. [31]



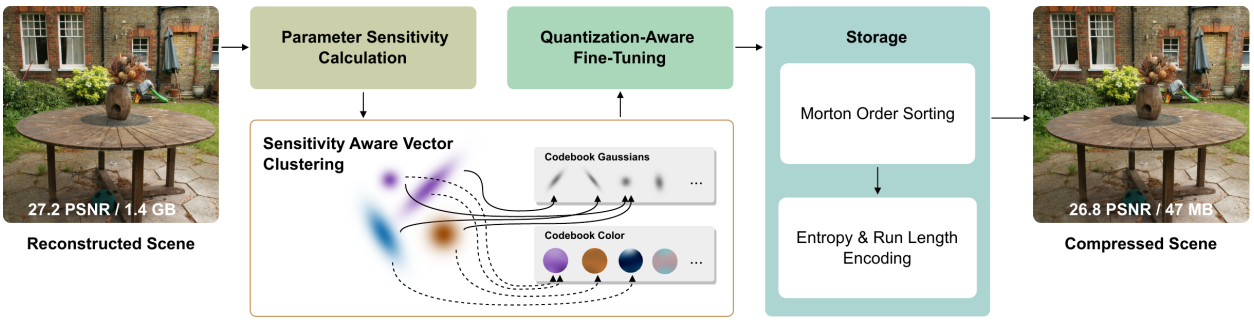
3.21 MesonGS: Post-training Compression of 3D Gaussians via Efficient Attribute Transformation

MesonGS employs universal Gaussian pruning by evaluating the importance of Gaussians through forward propagation, considering both view-dependent and view-independent features. It transforms rotation quaternions into Euler angles to reduce storage requirements and applies region adaptive hierarchical transform (RAHT) to reduce entropy in key attributes. Block quantization is performed on attribute channels by dividing them into multiple blocks and perform quantization for each block individually, using vector quantization for compressing less important attributes. Geometry is compressed using an octree, and all elements are packed with the LZ77 codec. A finetune scheme is implemented post-training to restore quality. [33]



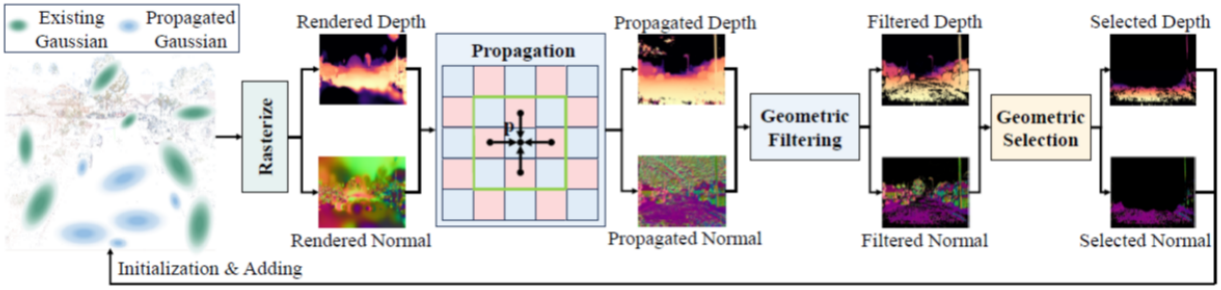
3.22 Compressed 3D Gaussian Splatting for Accelerated Novel View Synthesis

The authors propose a compressed 3D Gaussian splat representation consisting of three main steps: 1. sensitivity-aware clustering, where scene parameters are measured according to their contribution to the training images and encoded into compact codebooks via sensitivity-aware vector quantization; 2. quantization-aware fine-tuning, which recovers lost information by fine-tuning parameters at reduced bit-rates using quantization-aware training; and 3. entropy encoding, which exploits spatial coherence through entropy and run-length encoding by linearizing 3D Gaussians along a space-filling curve. Furthermore, a renderer for the compressed scenes utilizing GPU-based sorting and rasterization is proposed, enabling real-time novel view synthesis on low-end devices. [24]



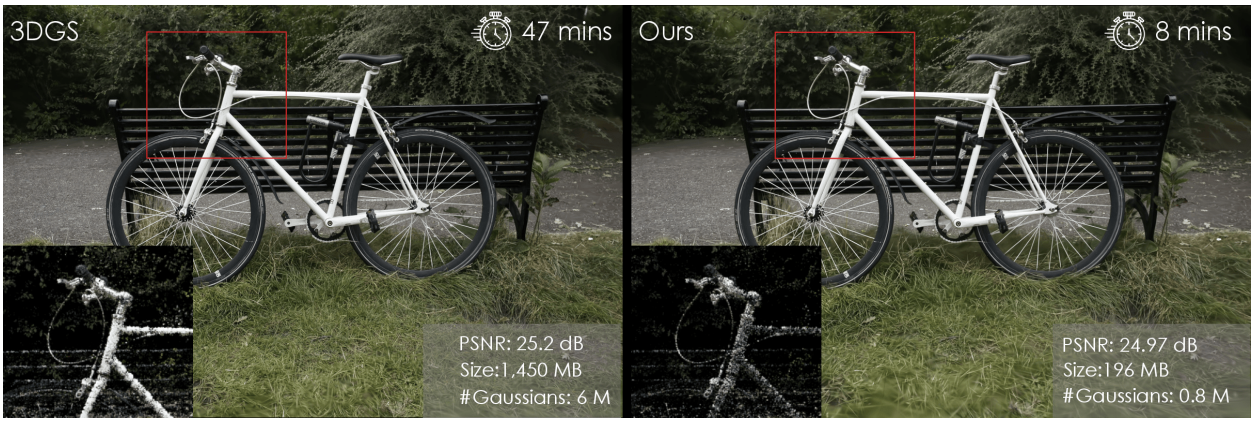
3.23 GaussianPro: 3D Gaussian Splatting with Progressive Propagation

This method generates depth and normal maps that guide the growth and adjustment of Gaussians. It employs patch matching to propagate depth and normal information from neighboring pixels to generate new values. Geometric filtering and selection then identify pixels needing additional Gaussians, which are initialized using the propagated information. It also introduces a planar loss to ensure Gaussians match real surfaces more closely. This method enforces consistency between the Gaussian's rendered normal and the propagated normal using L1 and angular loss. [4]



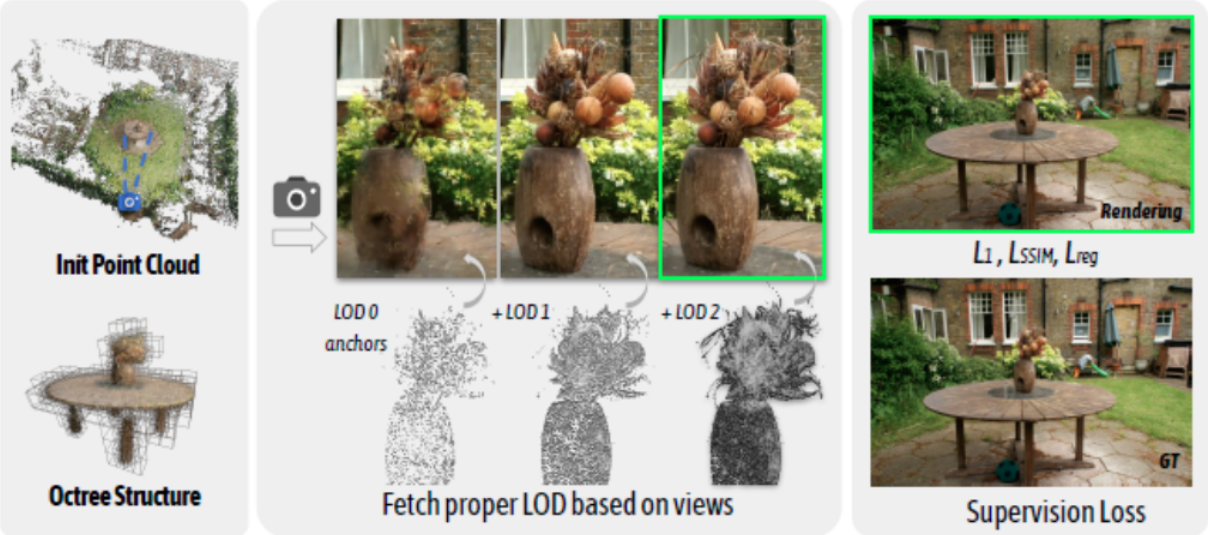
3.24 Taming 3DGS: High-Quality Radiance Fields with Limited Resources

This method employs a global scoring approach to guide the addition of Gaussians, ensuring efficient densification. Each Gaussian is assigned a score based on four factors: 1) gradient, 2) pixel coverage, 3) per-view saliency, and 4) core attributes like opacity, depth, and scale. Gaussians with the top B scores, where B is the desired number of new Gaussians, are then split or cloned to optimize the scene's representation. By calculating a composite score that reflects both the scene's structural complexity and visual importance, only the most critical areas are targeted for Gaussian splitting or cloning, resulting in more effective scene representation. [27]



3.25 Octree-GS: Towards Consistent Real-time Rendering with LOD-Structured 3D Gaussians

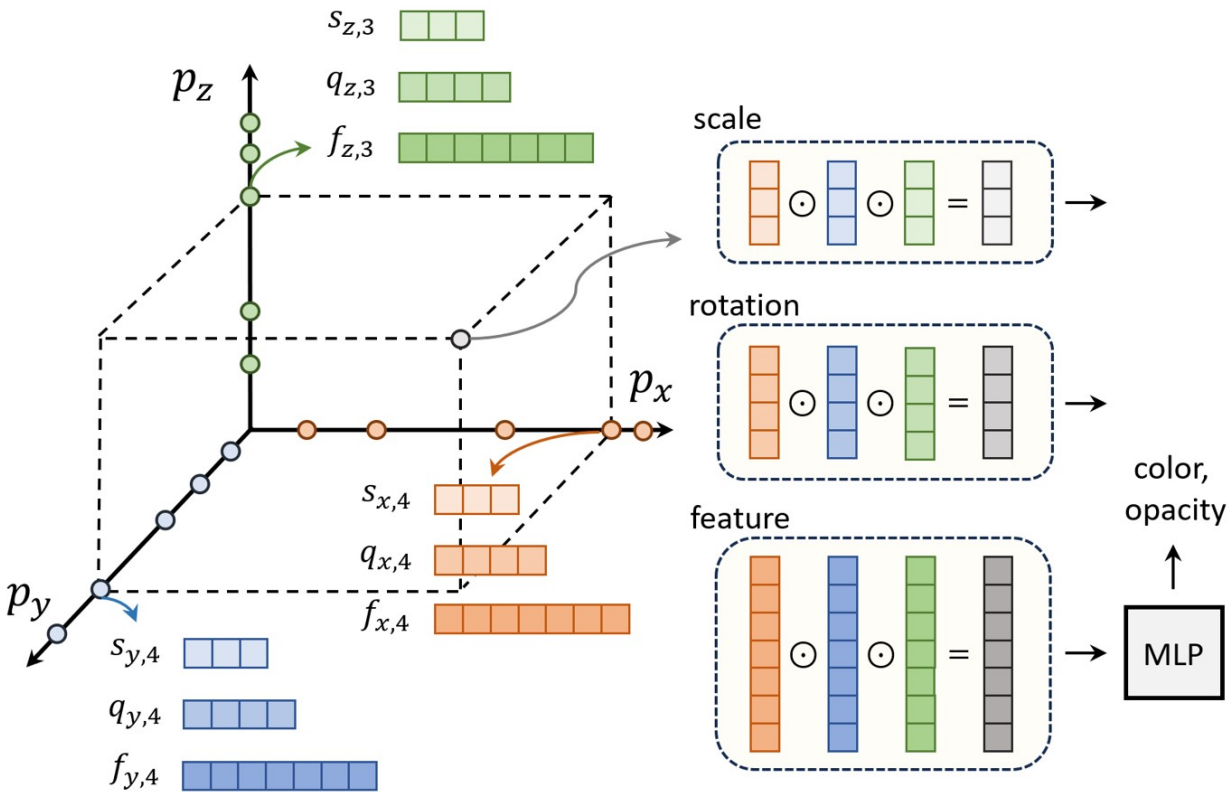
Octree-GS introduces an octree structure to 3D Gaussian splatting. Starting with a sparse point cloud, an octree is constructed for the bounded 3D space, where each level corresponds to a set of anchor Gaussians assigned to different levels of detail (LOD). This method selects the necessary LOD based on the observation view, gradually accumulating Gaussians from higher LODs for final rendering. The model is trained using standard image reconstruction and volume regularization losses. [26]



3.26 F-3DGS: Factorized Coordinates and Representations for 3D Gaussian Splatting

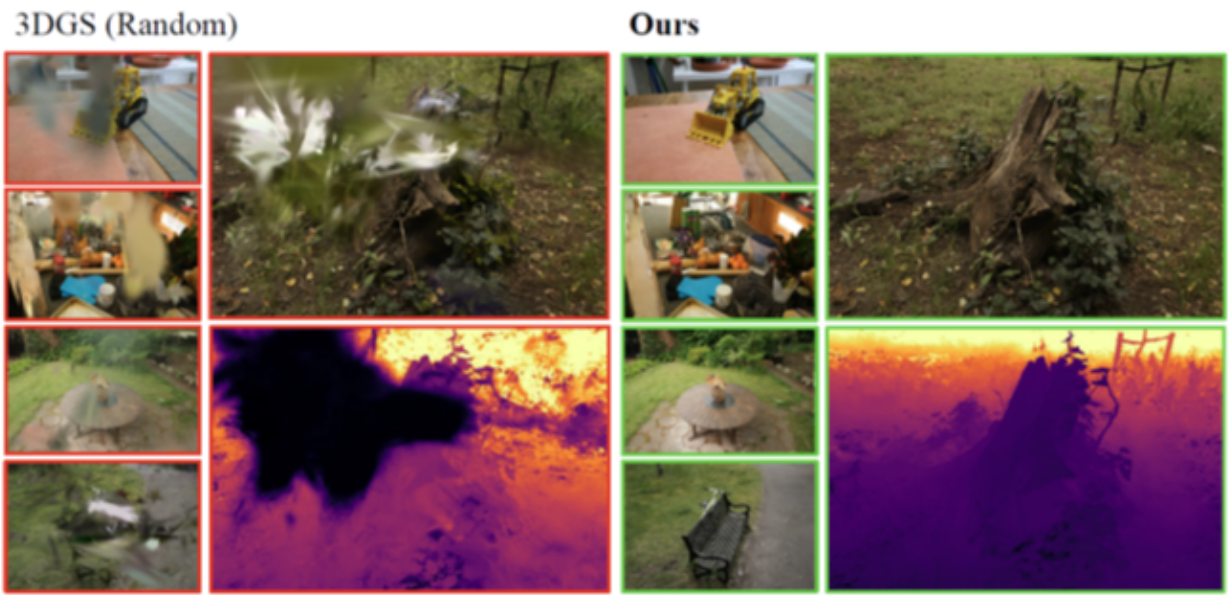
The paper introduces a novel 3D Gaussian compression method using structured coordinates and decomposed representations through factorization. Inspired by tensor or matrix factorization techniques, this method generates 3D coordinates via a tensor product of 1D or 2D coordinates, enhancing spatial efficiency. It extends factorization to include attributes like color, scale, rotation, and opacity, which compresses the model size while maintaining essential characteristics. A binary mask is employed to eliminate non-essential Gaussians, significantly accelerating training and rendering speeds.

Note: This paper is currently not included in the survey table because it shows unusually high results in the Tanks and Temples dataset and reports higher results for the original 3DGS than those in the original publication. This raises the possibility that their testing methods may differ from those used in other papers. [29]



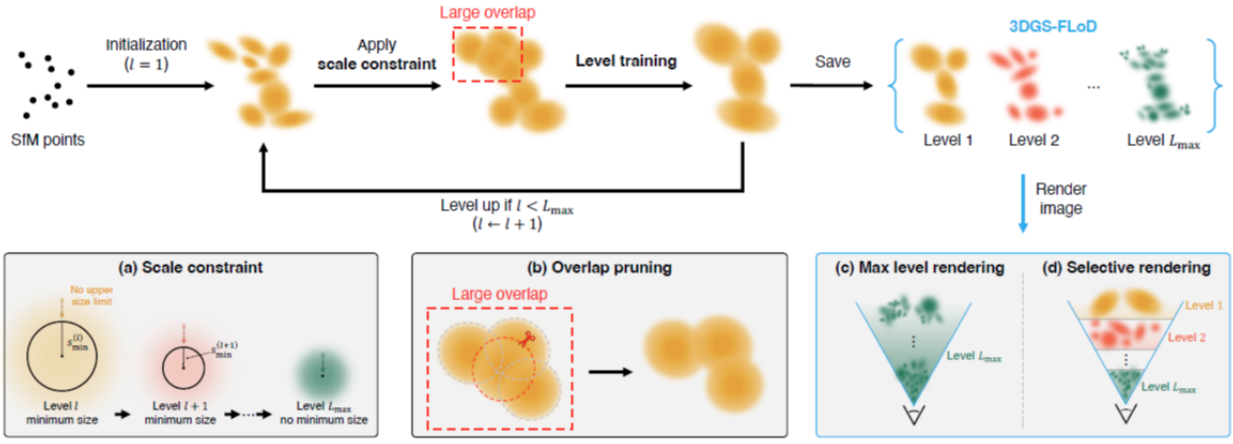
3.27 RAIN-GS: Relaxing Accurate Initialization Constraint for 3D Gaussian Splatting

This method introduces three strategies to enhance 3D Gaussian splatting. First, it employs Sparse-Large-Variance (SLV) initialization, which begins with a small number of Gaussians to capture the overall structure of the point cloud. Second, progressive Gaussian low-pass filtering encourages initial large-scale Gaussians to cover wider areas. Third, the adaptive bound-expanding split (ABE-Split) algorithm moves Gaussians beyond their initial bounds, encouraging them to explore a broader space for more accurate geometry reconstruction. [12]



3.28 FLoD: Integrating Flexible Level of Detail into 3D Gaussian Splatting for Customizable Rendering

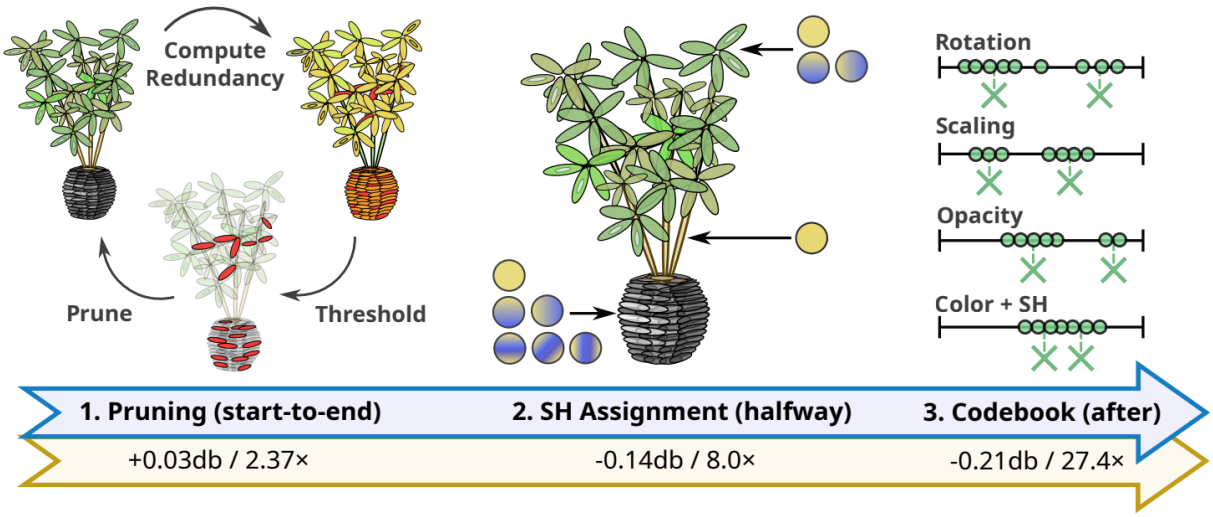
This method introduces a multi-level training approach to customize and reduce the size of 3D Gaussian splatting. Starting from Structure from Motion (SfM) points, training is performed level by level. At each level, (a) a scale constraint is applied to control Gaussian size, and (b) overlap pruning is used to reduce excessive Gaussian overlap. The final model supports maximum-level rendering for high-quality output or selective multi-level rendering for more efficient performance. [28]



3.29 Reducing the Memory Footprint of 3D Gaussian Splatting

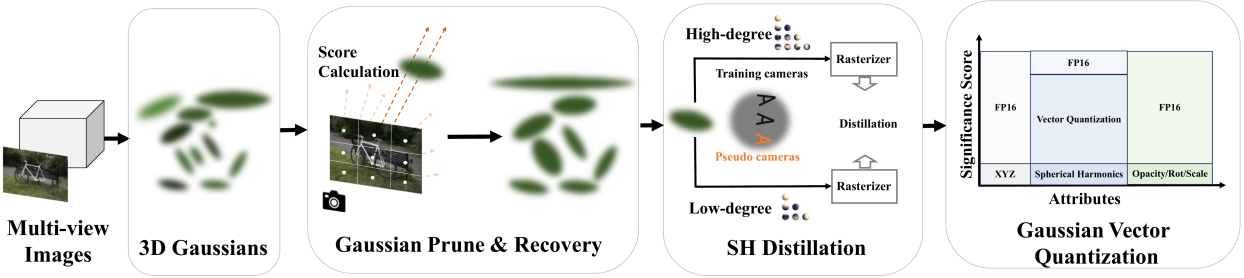
This approach addresses three main issues contributing to large storage sizes in 3D Gaussian Splatting (3DGS). To reduce the number of 3D Gaussian primitives, the authors introduce a scale- and resolution-aware redundant primitive removal method. This extends opacity-based pruning by incorporating a redundancy score to identify regions with many low-impact primitives. To mitigate storage size due to spherical harmonic coefficients, they propose adaptive adjustment of spherical harmonic (SH) bands. This involves evaluating color consistency across views and reducing higher-order SH bands when view-dependent effects are minimal. Additionally, recognizing the limited need for high dynamic range and precision for most primitive attributes,

they develop a codebook using K-means clustering and apply 16-bit half-float quantization to the remaining uncompressed floating point values. [25]



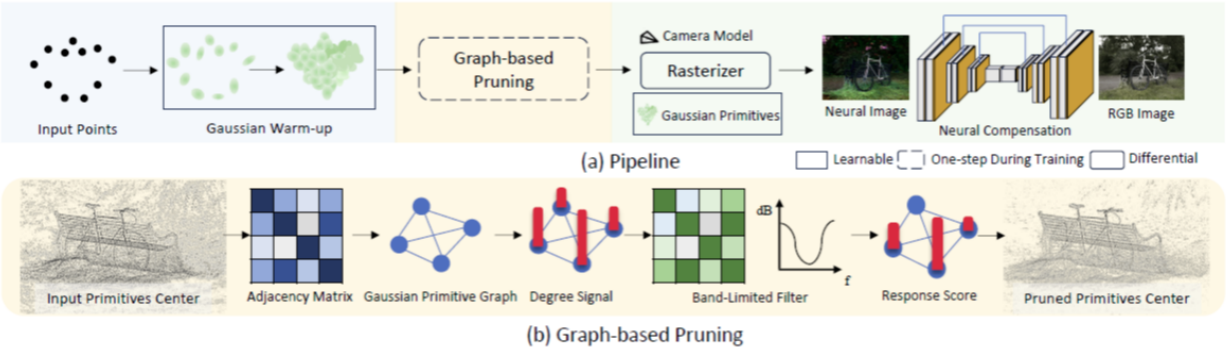
3.30 LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS

LightGaussian aims to transform 3D Gaussians to a more efficient and compact form, avoiding the scalablity issues that arrises from the large number of SfM (Structure from Motion) points for unbounded scenes. Inspired by Network Pruning, the method identifies Gaussians that minimally contribute to scene reconstruction and employs a pruning and recovery process, thereby efficiently reducing redundancy in Gaussian counts while maintaining visual effects. Additionally, LightGaussian utilizes knowledge distillation and pseudo-view augmentation to transfer spherical harmonics efficients to a lower degree. Furthermore, the authors propose a Gaussian Vector Quantization based on the global significance of Gaussians to quantize all redundant attributes, achieving lower bitwidth representations with minimal accuracy losses. [5]



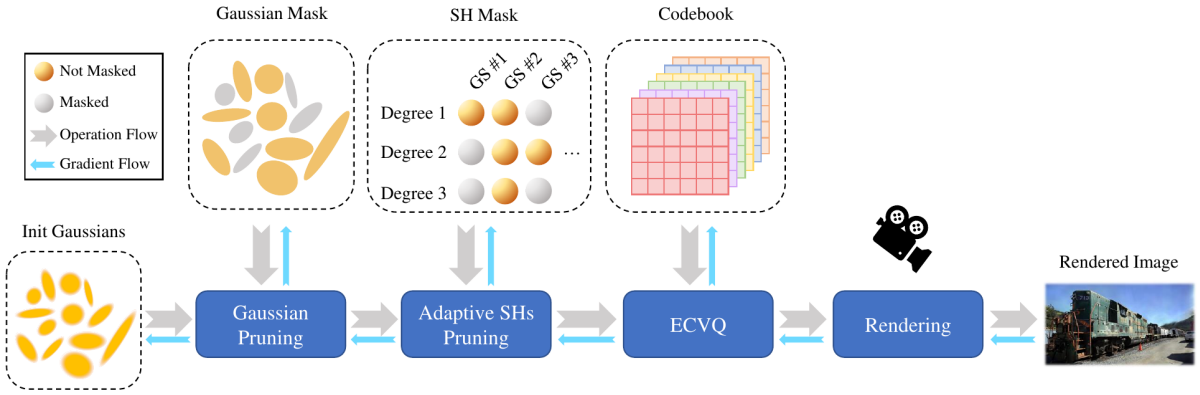
3.31 SUNDAE: Spectrally Pruned Gaussian Fields with Neural Compensation

This method combines graph-based pruning of Gaussian primitives with a convolutional neural network to retain high-frequency details. First, it "warms up" the 3D Gaussian field before applying a graph-based pruning strategy. The pruning step uses a graph built on the spatial relationships between the Gaussians and applies a band-limited graph filter to selectively down-sample, preserving important features. A convolutional neural network is then used to compensate for any losses caused by the pruning, ensuring that both high-frequency details and general low-frequency features are retained. [35]



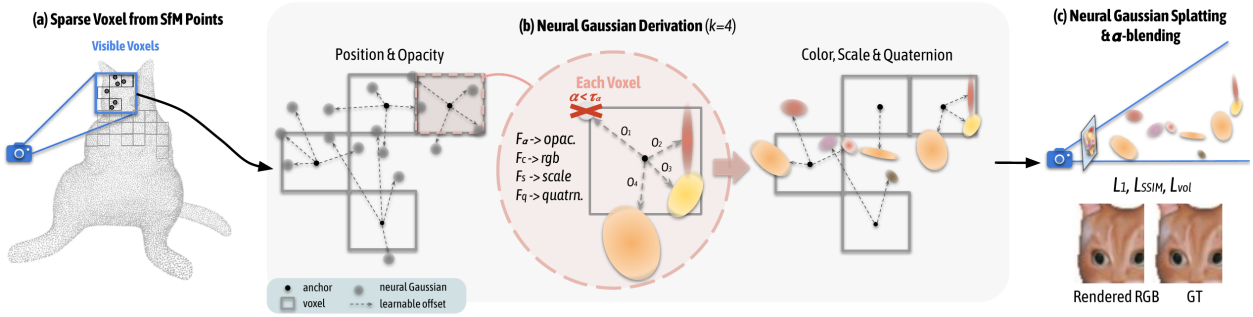
3.32 End-to-End Rate-Distortion Optimized 3D Gaussian Representation

This paper introduces RDO-Gaussian, an end-to-end Rate-Distortion Optimized 3D Gaussian representation. The authors achieve flexible, continuous rate control by formulating 3D Gaussian representation learning as a joint optimization of rate and distortion. Rate-distortion optimization is realized through dynamic pruning and entropy-constrained vector quantization (ECVQ). Gaussian pruning involves learning a mask to eliminate redundant Gaussians and adaptive SHs pruning assigns varying SH degrees to each Gaussian based on material and illumination needs. The covariance and color attributes are discretized through ECVQ, which performs vector quantization. [30]



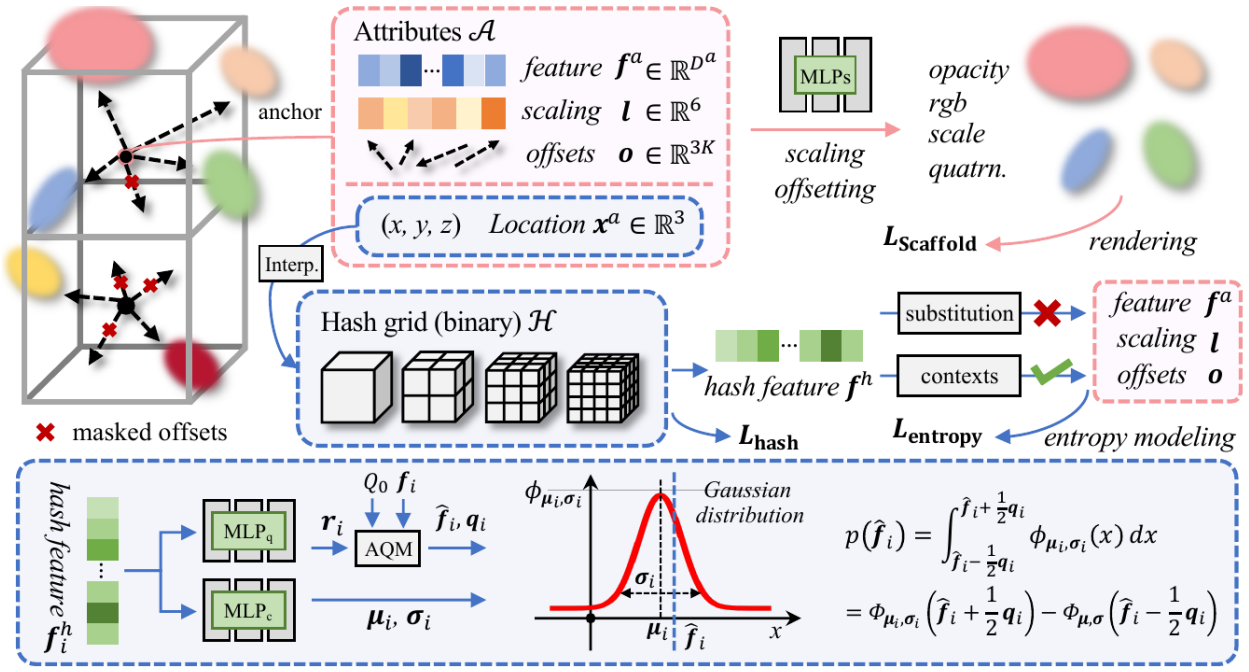
3.33 Scaffold-GS: Structured 3D Gaussians for View-Adaptive Rendering

Scaffold-GS introduces anchor points that leverage scene structure to guide the distribution of local 3D Gaussians. Attributes like opacity, color, rotation, and scale are dynamically predicted for Gaussians linked to each anchor within the viewing frustum, enabling adaptation to different viewing directions and distances. Initial anchor points are derived by voxelizing the sparse, irregular point cloud from Structure from Motion (SfM), forming a regular grid. To refine and grow the anchors, Gaussians are spatially quantized using voxels, with new anchors created at the centers of significant voxels, identified by their average gradient over N training steps. Random elimination and opacity-based pruning regulate anchor growth and refinement. [19]



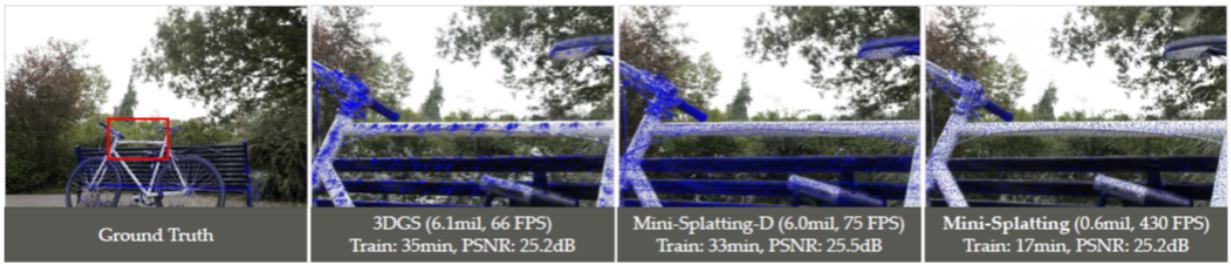
3.34 HAC: Hash-grid Assisted Context for 3D Gaussian Splatting Compression

The paper proposes a Hash-grid Assisted Context (HAC) framework for compressing 3D Gaussian Splatting (3DGS) models by leveraging the mutual information between attributes of unorganized 3D Gaussians (anchors) and hash grid features. Using Scaffold-GS as a base model, HAC queries the hash grid by anchor location to predict anchor attribute distributions for efficient entropy coding. The framework introduces an Adaptive Quantization Module (AQM) to dynamically adjust quantization step sizes. Furthermore, this method employs adaptive offset masking with learnable masks to eliminate invalid Gaussians and anchors, by leveraging the pruning strategy introduced by Compact3DGS and additionally removing anchors if all the attached offsets are pruned. [2]



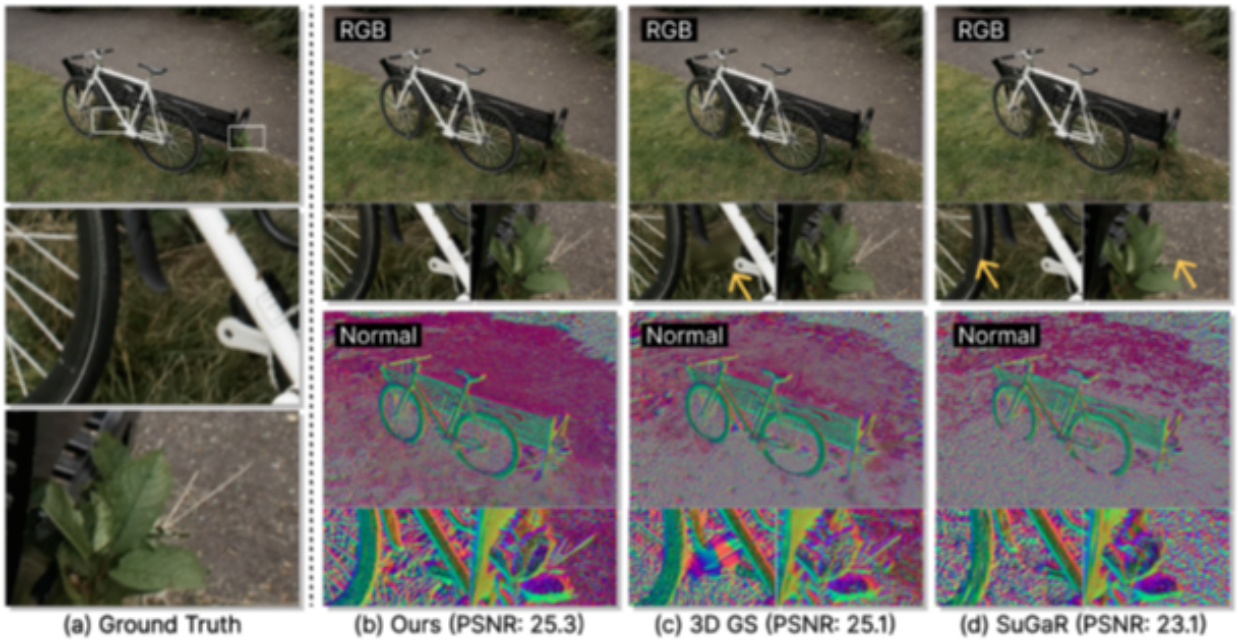
3.35 Mini-Splatting: Representing Scenes with a Constrained Number of Gaussians

Mini-Splatting enhances Gaussian distribution through Blur Split, which refines Gaussians in blurred regions, and Depth Reinitialization, which repositions Gaussians based on newly generated depth points, calculated from the mid-point of ray intersections with Gaussian ellipsoids, thus avoiding artifacts from alpha blending. For simplification, Intersection Preserving retains Gaussians with the greatest visual impact, while Sampling maintains geometric integrity and rendering quality, reducing complexity. [6]



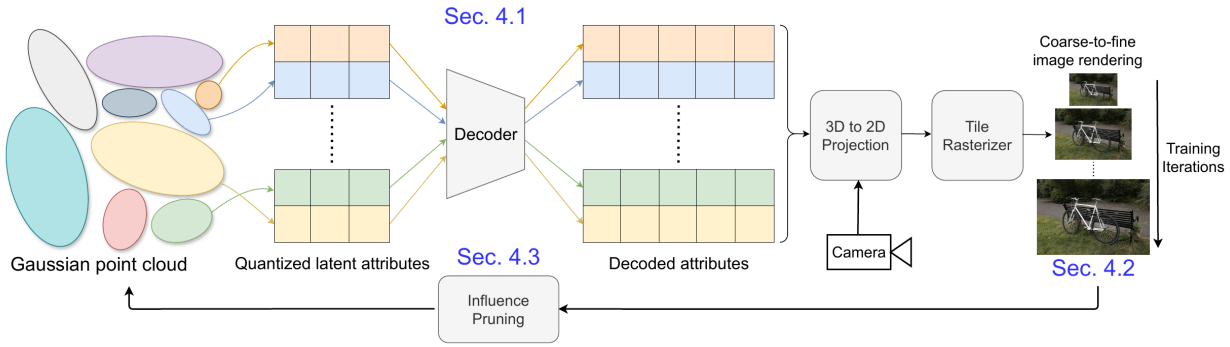
3.36 AtomGS: Atomizing Gaussian Splatting for High-Fidelity Radiance Field

This method prioritizes fine details through Atom Gaussians, which are isotropic and uniformly sized to align closely with the scene's geometry, while large Gaussians are merged to cover smooth surfaces. In addition, Geometry-Guided Optimization uses an Edge-Aware Normal Loss and multi-scale SSIM to maintain geometric accuracy. The Edge-Aware Normal Loss is calculated as the product of the normal map, derived from the pre-optimized 3DGS, and the edge map, which is derived from the gradient magnitude of the ground truth RGB image. [18]



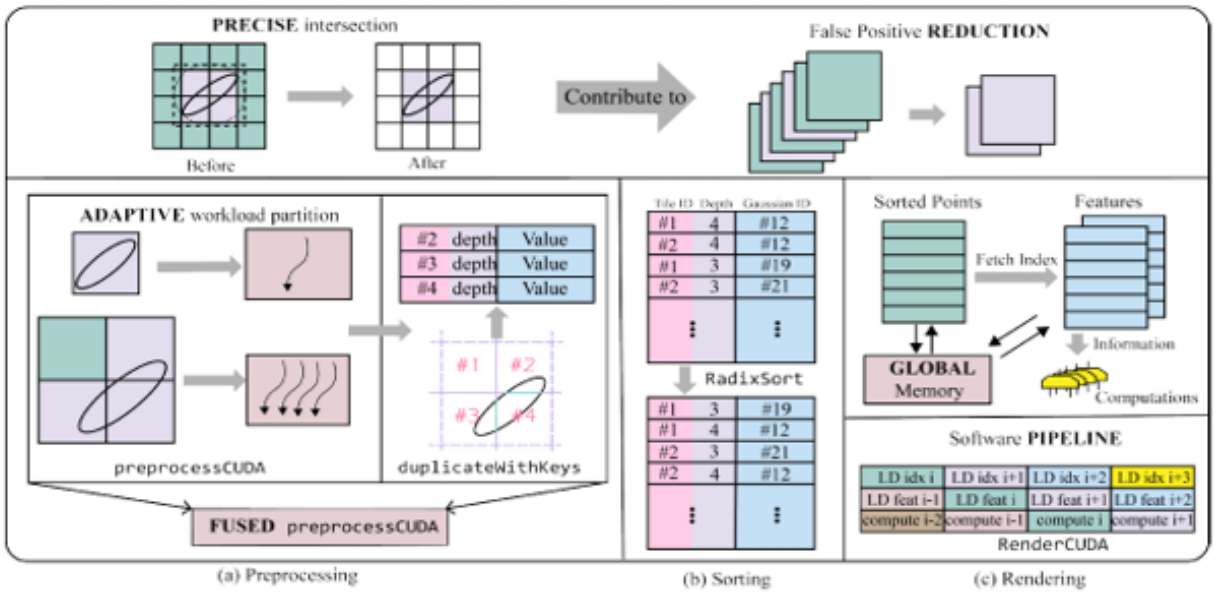
3.37 EAGLES: Efficient Accelerated 3D Gaussians with Lightweight EncodingS

The authors of this approach observed that in 3DGS, the color and rotation attributes account for over 80% of memory usage; thus, they propose compressing these attributes via a latent quantization framework. Additionally, they quantize the opacity coefficients of the Gaussians, improving optimization and resulting in fewer floaters or visual artifacts in novel view reconstructions. To reduce the number of redundant Gaussians resulting from frequent densification (via cloning and splitting), the approach employs a pruning stage to identify and remove Gaussians with minimal influence on the full reconstruction. For this, an influence metric is introduced, which considers both opacity and transmittance. [9]



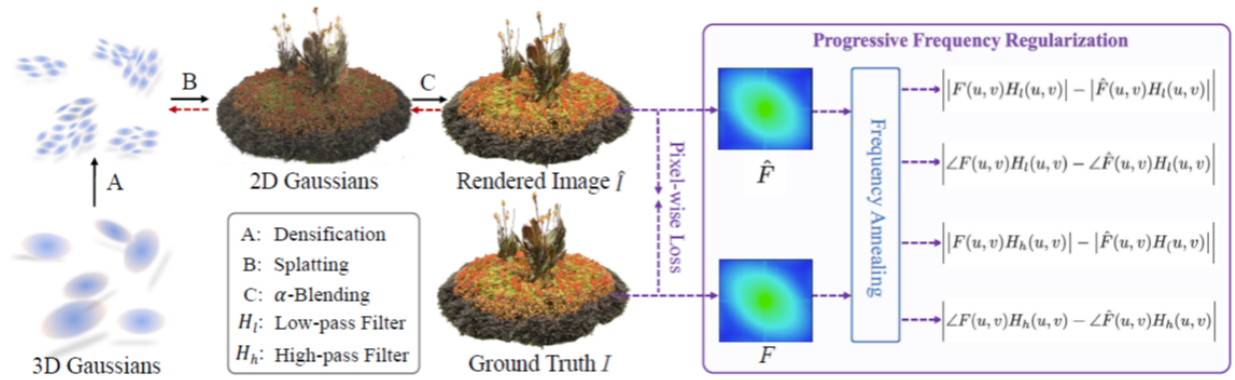
3.38 FlashGS: Efficient 3D Gaussian Splatting for Large-scale and High-resolution Rendering

The method reduces computational overhead by addressing inefficiencies in direct intersection calculations for elongated ellipses. It employs a two-stage filtering process: (1) a tightly tangent bounding box is used to set a coarse range, and (2) tiles intersecting the bounding box are checked for ellipse intersections. Additionally, FlashGS reduces global memory queries by employing a software pipelining approach that overlaps instruction dispatch, optimizing data retrieval and rendering efficiency. [8]



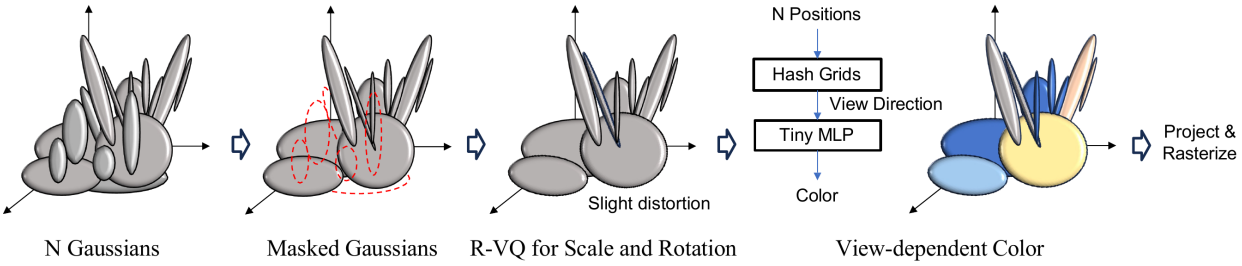
3.39 FreGS: 3D Gaussian Splatting with Progressive Frequency Regularization

This approach is designed to address over-reconstruction from a frequency-based perspective. It minimizes the discrepancy between the frequency spectra of rendered images and the corresponding ground truth. This method explicitly regularizes amplitude and phase differences within Fourier space to achieve more accurate results. [36]



3.40 Compact 3D Gaussian Representation for Radiance Field

This approach introduces a Gaussian volume mask to prune non-essential Gaussians and a compact attribute representation for both view-dependent color and geometric attributes. The volume-based masking strategy combines opacity and scale to selectively remove redundant Gaussians. For color attribute compression, spatial redundancy is exploited by incorporating a grid-based (Instant-NGP) neural field, allowing efficient representation of view-dependent colors without storing attributes per Gaussian. Given the limited variation in scale and rotation, geometric attribute compression employs a compact codebook-based representation to identify and reuse similar geometries across the scene. Additionally, the authors propose quantization and entropy-coding as post-processing steps for further compression. [15]



3.41 gsplat

This approach leverages 3D Gaussian Splatting as Markov Chain Monte Carlo (3DGS-MCMC), interpreting the training process of positioning and optimizing Gaussians as a sampling procedure rather than minimizing a predefined loss function.

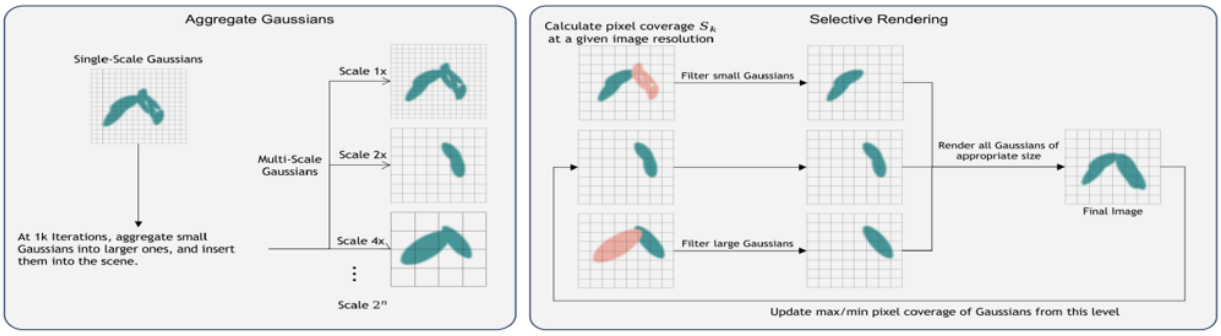
Additionally, it incorporates compression techniques derived from the Morgenstern et al. paper, which organizes the parameters of 3DGS in a 2D grid, capitalizing on perceptual redundancies found in natural scenes, thereby significantly reducing storage requirements.

Further compression is achieved by applying methods from Making Gaussian Splats more smaller, which reduces the size of Gaussian splats by clustering spherical harmonics into discrete elements and storing them as FP16 values.

This technique is implemented in gsplat, an open-source library designed for CUDA-accelerated differentiable rasterization of 3D Gaussians, equipped with Python bindings. [11]

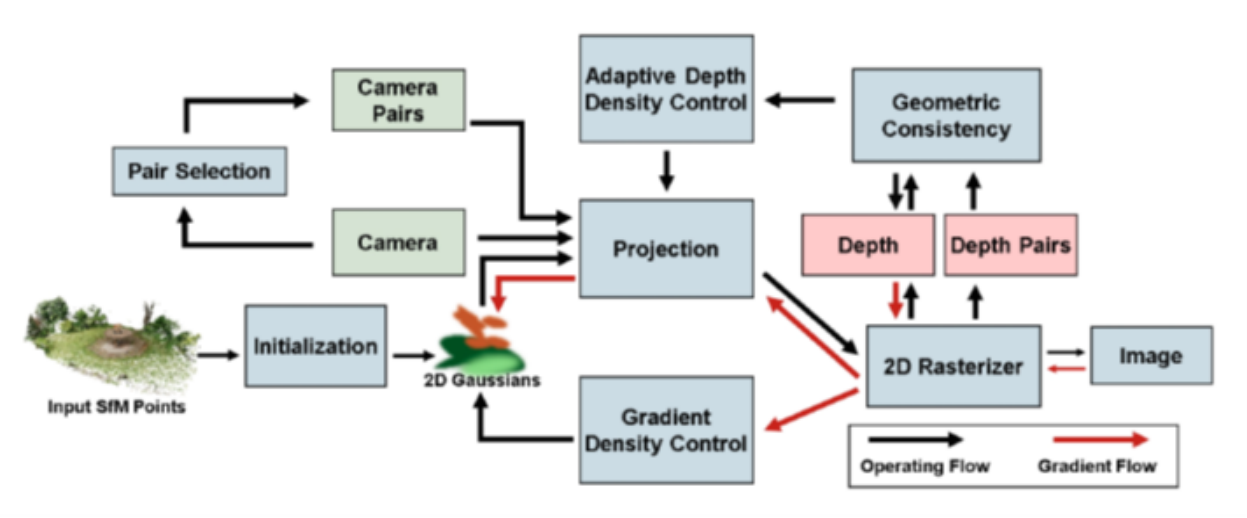
3.42 Multi-Scale 3D Gaussian Splatting for Anti-Aliased Rendering

This method aggregates small Gaussians below a size threshold during early training, enlarging and inserting them into the scene at various resolution scales. These multi-scale Gaussians are then selected for rendering based on their "pixel coverage" at the current resolution. During rendering, only Gaussians with pixel coverage between adaptively selected minimum and maximum are selected. These pixel-coverage values are dynamically updated based on the aggregation results from the first stage. [34]



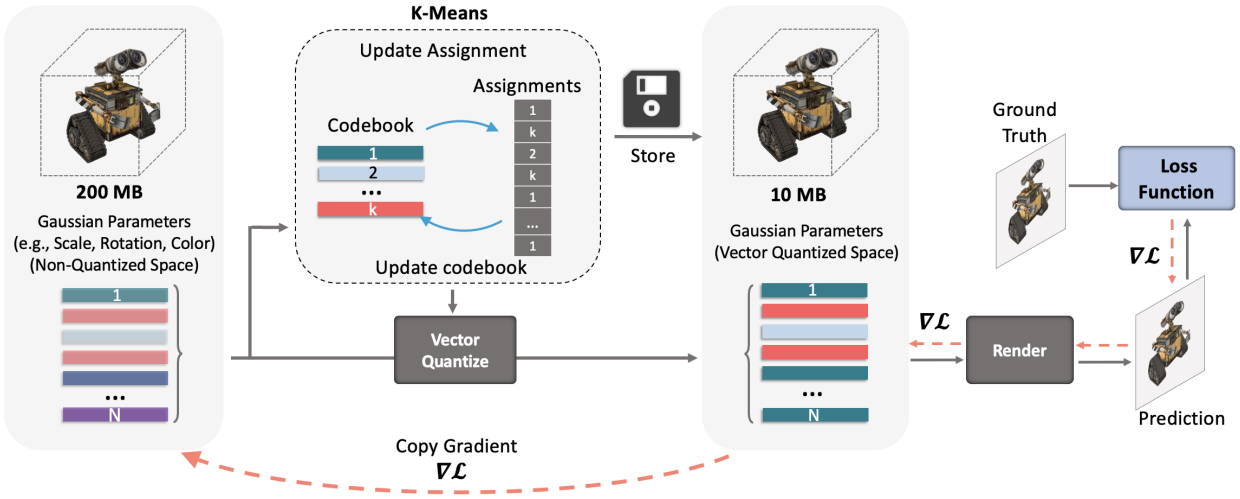
3.43 MVG-Splatting: Multi-View Guided Gaussian Splatting with Adaptive Quantile-Based Geometric Consistency Densification

This method introduces adaptive depth density control to enhance 2D GS. 2D GS renders depth maps. In this method, the render depth maps are dynamically divided into near, mid, and far regions based on Kernel Density Estimation (KDE) and Fast Fourier Transform (FFT), focusing densification in under-reconstructed near and far areas. Geometric consistency checks and depth projection ensure adaptive densification without over-reconstruction, particularly in these critical regions. [17]



3.44 Compact3D: Compressing Gaussian Splat Radiance Field Models with Vector Quantization

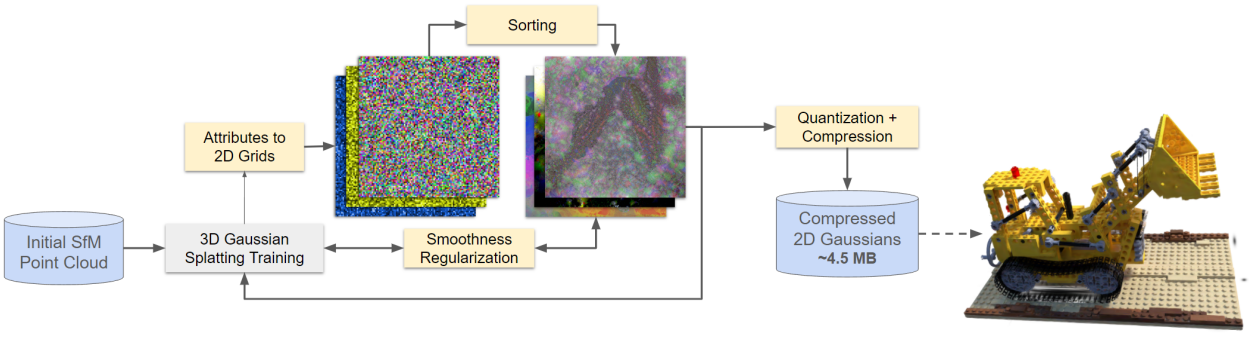
This approach introduces a vector quantization method based on the K-means algorithm to quantize the Gaussian parameters in 3D Gaussian splatting, as many Gaussians may share similar parameters. Only a small codebook is stored along with the index of the code for each Gaussian, resulting in a large reduction in the storage of the learned radiance fields and a reduction of the memory footprint at rendering time. Additionally, the indices are further compressed by sorting the Gaussians based on one of the quantized parameters and storing the indices using a method similar to Run-Length-Encoding (RLE). To reduce the number of Gaussians, this method applies a regularizer to encourage zero opacity, before pruning Gaussians with opacity smaller than a threshold. [23]



3.45 Compact 3D Scene Representation via Self-Organizing Gaussian Grids

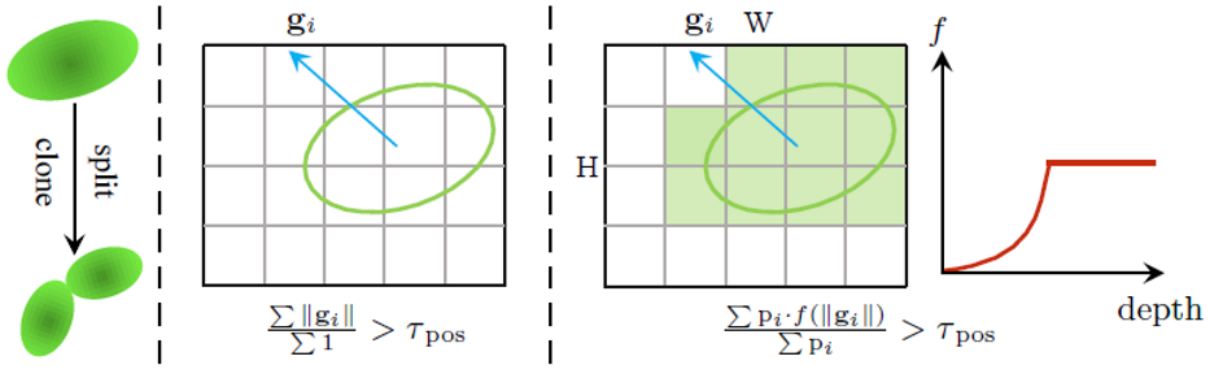
Compressing 3D data is challenging, but many effective solutions exist for compressing 2D data (such as images). The authors propose a new method to organize 3DGS parameters into a 2D grid, drastically reducing storage requirements without compromising visual quality. This organization exploits perceptual redundancies in natural scenes. They introduce a highly parallel sorting algorithm, PLAS, which arranges Gaussian parameters into a 2D grid, maintaining local neighborhood structure and ensuring smoothness.

This solution is particularly innovative because no existing method efficiently handles a 2D grid with millions of points. During training, a smoothness loss is applied to enforce local smoothness in the 2D grid, enhancing the compressibility of the data. The key insight is that smoothness needs to be enforced during training to enable efficient compression. [22]



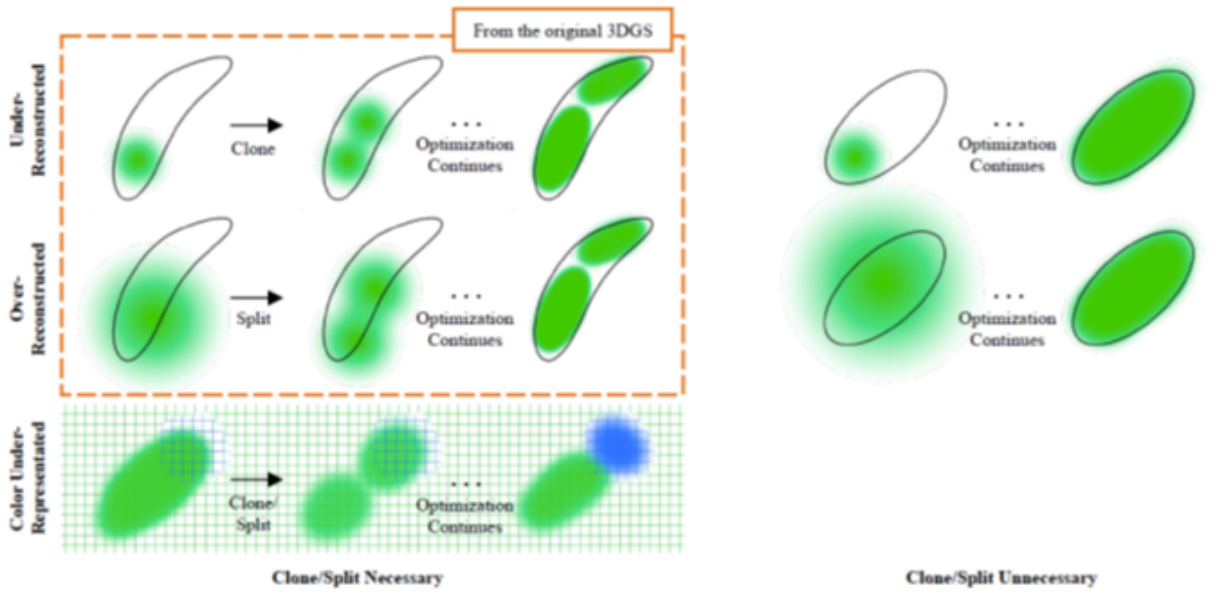
3.46 Pixel-GS: Density Control with Pixel-aware Gradient for 3D Gaussian Splatting

The Pixel-GS method introduces a pixel-aware gradient to address under-reconstruction and needle-like artifacts in 3DGS. This gradient guides densification when the averaged scale gradient across multiple views exceeds a threshold, unlike the original method that only considers a single view. Additionally, Pixel-GS proposes a depth-dependent gradient scaling strategy to balance the influence of Gaussians on affected areas, as near-view Gaussians often receive an excessive number of gradients due to affecting too many pixels. [37]



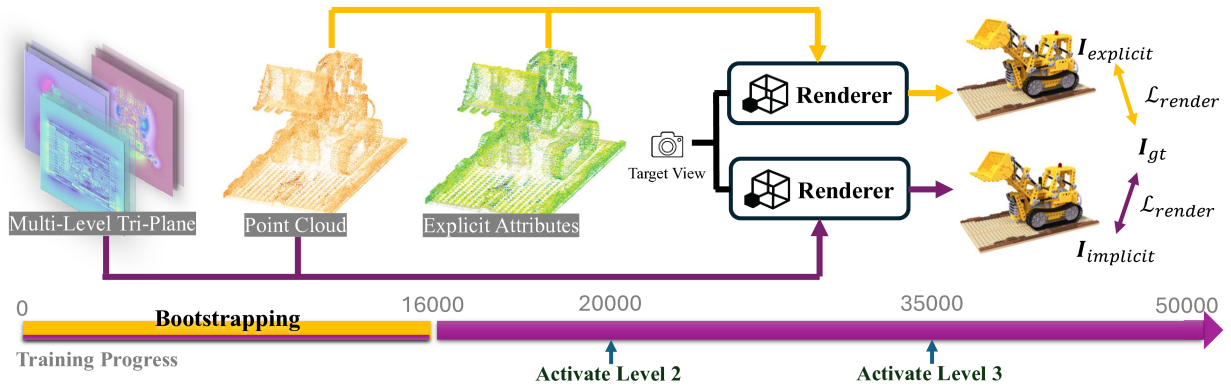
3.47 Color-cued Efficient Densification Method for 3D Gaussian Splatting

This method introduces a simple yet effective modification to the densification process in the original 3D Gaussian Splatting (3DGS). It leverages the view-independent (0th) spherical harmonics (SH) coefficient gradient to better assess color cues for densification, while using the 2D position gradient more coarsely to refine areas where structure-from-motion (SfM) struggles to capture fine structures. [13]



3.48 Implicit Gaussian Splatting with Efficient Multi-Level Tri-Plane Representation

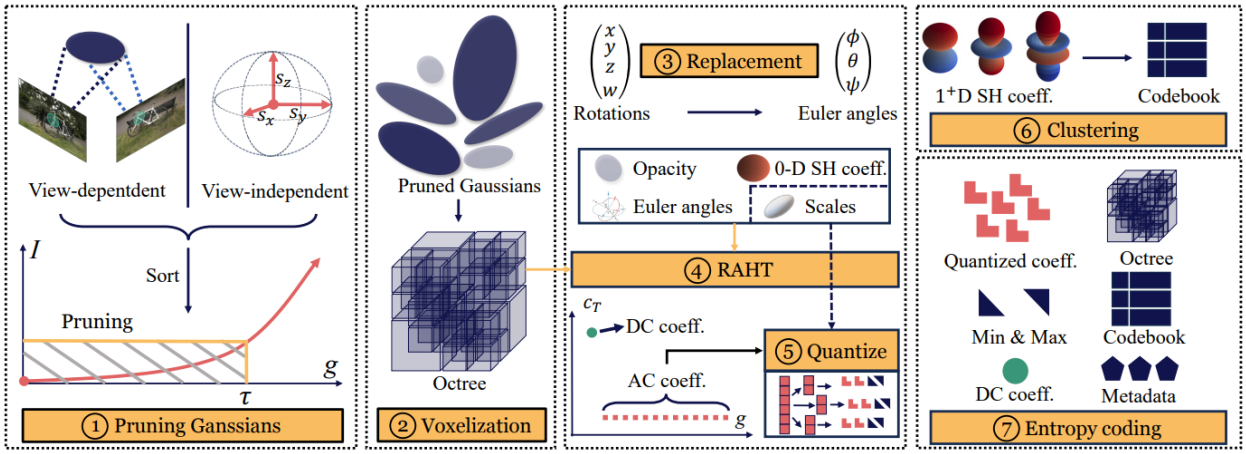
This method introduces a hybrid representation for splatting-based radiance fields, where Gaussian primitives are separated into explicit point cloud and implicit attribute features. The attribute features are encoded using a multi-resolution multi-level tri-plane architecture integrated with a residual-based rendering pipeline. It employs a level-based progressive training scheme for joint optimization of point clouds and tri-planes, starting with coarse attributes and refining them with higher-level details. Spatial regularization and a bootstrapping scheme are applied to enhance the consistency and stability of the Gaussian attributes during training. [31]



3.49 MesonGS: Post-training Compression of 3D Gaussians via Efficient Attribute Transformation

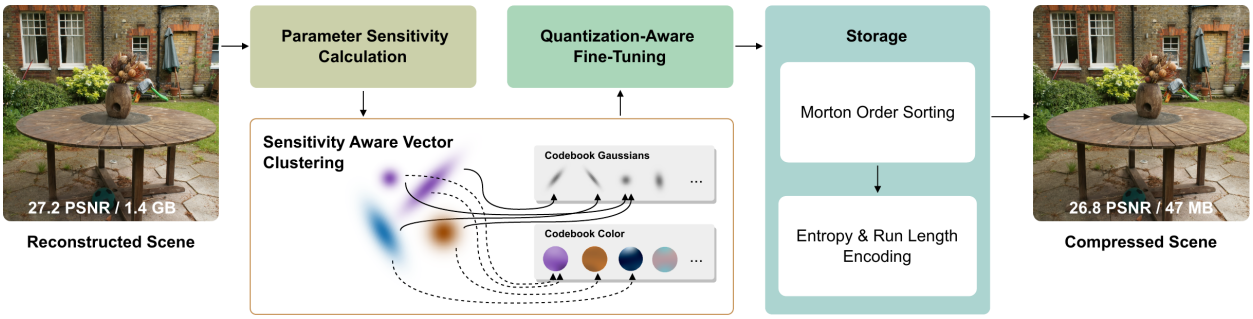
MesonGS employs universal Gaussian pruning by evaluating the importance of Gaussians through forward propagation, considering both view-dependent and view-independent features. It transforms rotation quaternions into Euler angles to reduce storage requirements and applies region adaptive hierarchical transform (RAHT) to reduce entropy in key attributes. Block quantization is performed on attribute channels by dividing them into multiple blocks and perform quantization for each block individually, using vector quantization for compressing less important attributes. Geometry is compressed using an octree, and all elements

are packed with the LZ77 codec. A finetune scheme is implemented post-training to restore quality. [33]



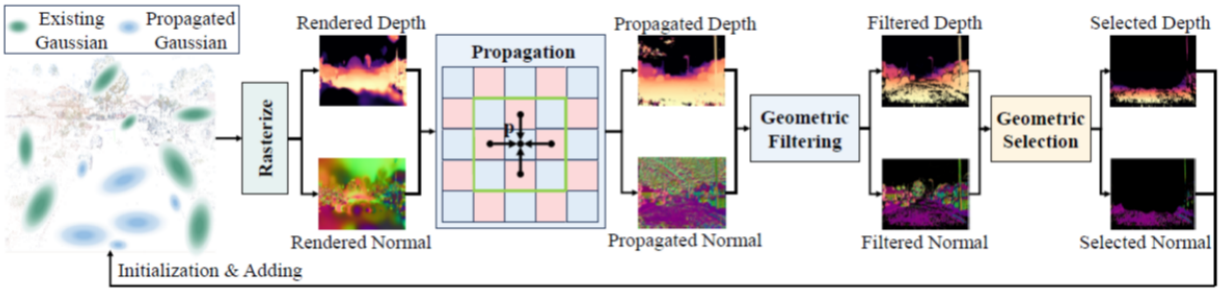
3.50 Compressed 3D Gaussian Splatting for Accelerated Novel View Synthesis

The authors propose a compressed 3D Gaussian splat representation consisting of three main steps: 1. sensitivity-aware clustering, where scene parameters are measured according to their contribution to the training images and encoded into compact codebooks via sensitivity-aware vector quantization; 2. quantization-aware fine-tuning, which recovers lost information by fine-tuning parameters at reduced bit-rates using quantization-aware training; and 3. entropy encoding, which exploits spatial coherence through entropy and run-length encoding by linearizing 3D Gaussians along a space-filling curve. Furthermore, a renderer for the compressed scenes utilizing GPU-based sorting and rasterization is proposed, enabling real-time novel view synthesis on low-end devices. [24]



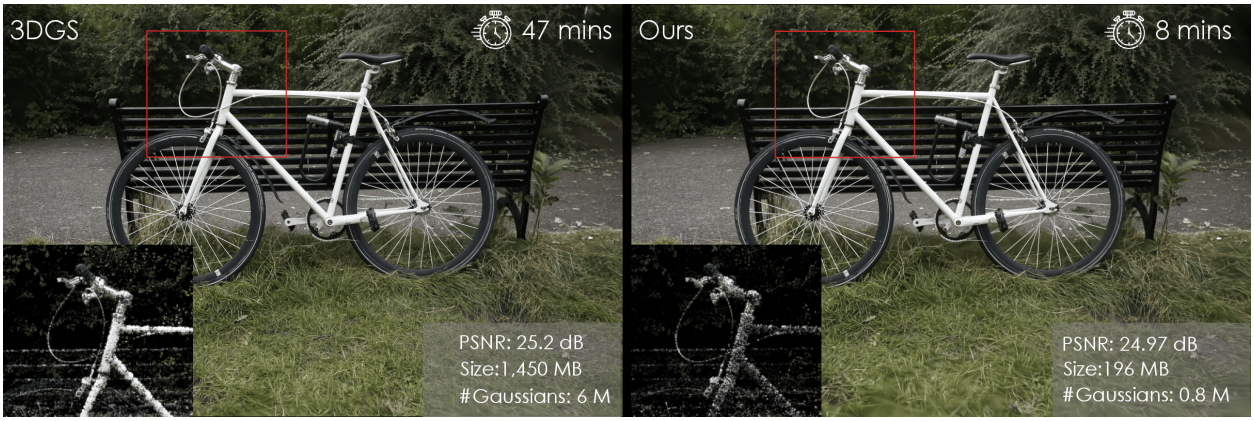
3.51 GaussianPro: 3D Gaussian Splatting with Progressive Propagation

This method generates depth and normal maps that guide the growth and adjustment of Gaussians. It employs patch matching to propagate depth and normal information from neighboring pixels to generate new values. Geometric filtering and selection then identify pixels needing additional Gaussians, which are initialized using the propagated information. It also introduces a planar loss to ensure Gaussians match real surfaces more closely. This method enforces consistency between the Gaussian's rendered normal and the propagated normal using L1 and angular loss. [4]



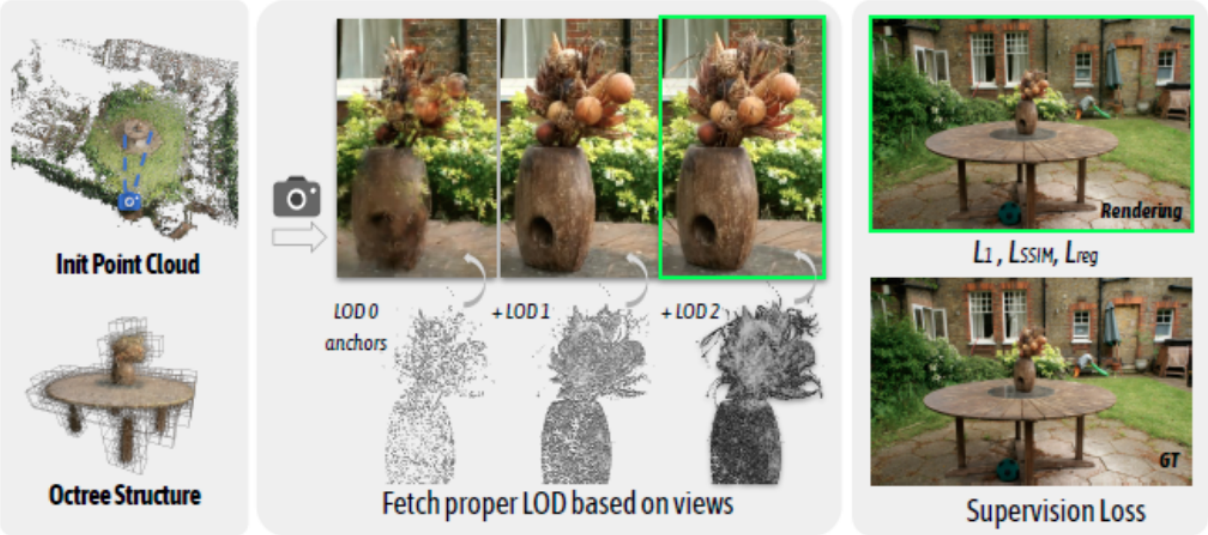
3.52 Taming 3DGS: High-Quality Radiance Fields with Limited Resources

This method employs a global scoring approach to guide the addition of Gaussians, ensuring efficient densification. Each Gaussian is assigned a score based on four factors: 1) gradient, 2) pixel coverage, 3) per-view saliency, and 4) core attributes like opacity, depth, and scale. Gaussians with the top B scores, where B is the desired number of new Gaussians, are then split or cloned to optimize the scene's representation. By calculating a composite score that reflects both the scene's structural complexity and visual importance, only the most critical areas are targeted for Gaussian splitting or cloning, resulting in more effective scene representation. [27]



3.53 Octree-GS: Towards Consistent Real-time Rendering with LOD-Structured 3D Gaussians

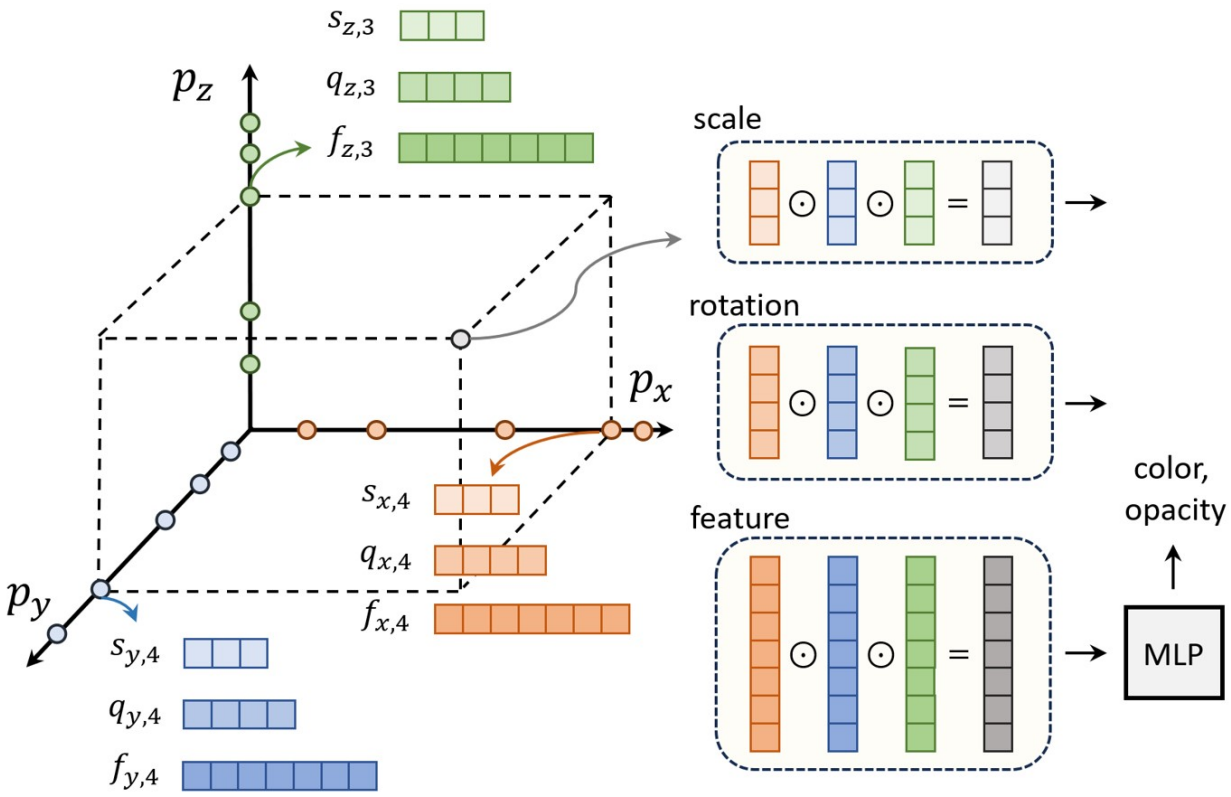
Octree-GS introduces an octree structure to 3D Gaussian splatting. Starting with a sparse point cloud, an octree is constructed for the bounded 3D space, where each level corresponds to a set of anchor Gaussians assigned to different levels of detail (LOD). This method selects the necessary LOD based on the observation view, gradually accumulating Gaussians from higher LODs for final rendering. The model is trained using standard image reconstruction and volume regularization losses. [26]



3.54 F-3DGS: Factorized Coordinates and Representations for 3D Gaussian Splatting

The paper introduces a novel 3D Gaussian compression method using structured coordinates and decomposed representations through factorization. Inspired by tensor or matrix factorization techniques, this method generates 3D coordinates via a tensor product of 1D or 2D coordinates, enhancing spatial efficiency. It extends factorization to include attributes like color, scale, rotation, and opacity, which compresses the model size while maintaining essential characteristics. A binary mask is employed to eliminate non-essential Gaussians, significantly accelerating training and rendering speeds.

Note: This paper is currently not included in the survey table because it shows unusually high results in the Tanks and Temples dataset and reports higher results for the original 3DGS than those in the original publication. This raises the possibility that their testing methods may differ from those used in other papers. [29]

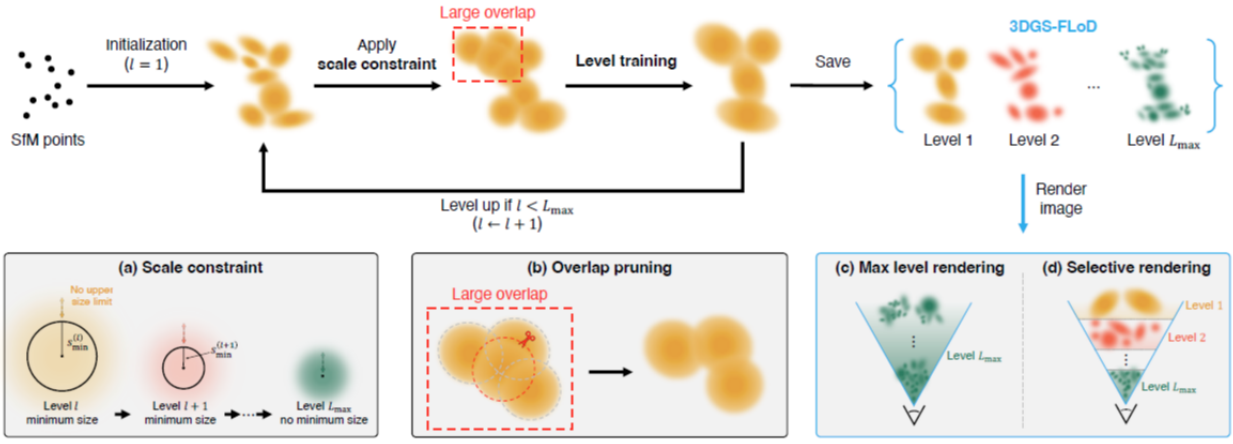


3.55 RAIN-GS: Relaxing Accurate Initialization Constraint for 3D Gaussian Splatting

This method introduces three strategies to enhance 3D Gaussian splatting. First, it employs Sparse-Large-Variance (SLV) initialization, which begins with a small number of Gaussians to capture the overall structure of the point cloud. Second, progressive Gaussian low-pass filtering encourages initial large-scale Gaussians to cover wider areas. Third, the adaptive bound-expanding split (ABE-Split) algorithm moves Gaussians beyond their initial bounds, encouraging them to explore a broader space for more accurate geometry reconstruction. [12]

3.56 FLoD: Integrating Flexible Level of Detail into 3D Gaussian Splatting for Customizable Rendering

This method introduces a multi-level training approach to customize and reduce the size of 3D Gaussian splatting. Starting from Structure from Motion (SfM) points, training is performed level by level. At each level, (a) a scale constraint is applied to control Gaussian size, and (b) overlap pruning is used to reduce excessive Gaussian overlap. The final model supports maximum-level rendering for high-quality output or selective multi-level rendering for more efficient performance. [28]



References

[1] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[2] Y. Chen, Q. Wu, J. Cai, M. Harandi, and W. Lin. Hac: Hash-grid assisted context for 3d gaussian splatting compression, 2024, 2403.14530.

[3] Y. Chen, Q. Wu, M. Harandi, and J. Cai. How far can we compress instant-ngp-based nerf? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20321–20330, 2024.

[4] K. Cheng, X. Long, K. Yang, Y. Yao, W. Yin, Y. Ma, W. Wang, and X. Chen. Gaussianpro: 3d gaussian splatting with progressive propagation, 2024, 2402.14650.

[5] Z. Fan, K. Wang, K. Wen, Z. Zhu, D. Xu, and Z. Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps, 2024, 2311.17245.

[6] G. Fang and B. Wang. Mini-splatting: Representing scenes with a constrained number of gaussians, 2024, 2403.14166.

[7] B. Fei, J. Xu, R. Zhang, Q. Zhou, W. Yang, and Y. He. 3d gaussian splatting as new era: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 2024.

[8] G. Feng, S. Chen, R. Fu, Z. Liao, Y. Wang, T. Liu, Z. Pei, H. Li, X. Zhang, and B. Dai. Flashgs: Efficient 3d gaussian splatting for large-scale and high-resolution rendering, 2024, 2408.07967.

[9] S. Girish, K. Gupta, and A. Shrivastava. Eagles: Efficient accelerated 3d gaussians with lightweight encodings, 2024, 2312.04564.

[10] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018.

[11] J. Hu, R. Li, V. Ye, and A. Kanazawa. gsplat compression, 2024.

[12] J. Jung, J. Han, H. An, J. Kang, S. Park, and S. Kim. Relaxing accurate initialization constraint for 3d gaussian splatting, 2024, 2403.09413.

[13] S. Kim, K. Lee, and Y. Lee. Color-cued efficient densification method for 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 775–783, June 2024.

[14] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017.

[15] J. C. Lee, D. Rho, X. Sun, J. H. Ko, and E. Park. Compact 3d gaussian representation for radiance field, 2024.

[16] L. Li, Z. Shen, Z. Wang, L. Shen, and L. Bo. Compressing volumetric radiance fields to 1 mb. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4222–4231, 2023.

[17] Z. Li, S. Yao, Y. Chu, A. F. Garcia-Fernandez, Y. Yue, E. G. Lim, and X. Zhu. Mvg-splatting: Multi-view guided gaussian splatting with adaptive quantile-based geometric consistency densification, 2024, 2407.11840.

[18] R. Liu, R. Xu, Y. Hu, M. Chen, and A. Feng. Atomgs: Atomizing gaussian splatting for high-fidelity radiance field, 2024, 2405.12369.

[19] T. Lu, M. Yu, L. Xu, Y. Xiangli, L. Wang, D. Lin, and B. Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering, 2024.

[20] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[21] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[22] W. Morgenstern, F. Barthel, A. Hilsmann, and P. Eisert. Compact 3d scene representation via self-organizing gaussian grids, 2024, 2312.13299.

[23] K. Navaneet, K. P. Meibodi, S. A. Koohpayegani, and H. Pirsiavash. Compact3d: Compressing gaussian splat radiance field models with vector quantization, 2024, 2311.18159.

[24] S. Niedermayr, J. Stumpfegger, and R. Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis, 2024.

[25] P. Papantonakis, G. Kopanas, B. Kerbl, A. Lanvin, and G. Drettakis. Reducing the memory footprint of 3d gaussian splatting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 7(1):1–17, May 2024.

[26] K. Ren, L. Jiang, T. Lu, M. Yu, L. Xu, Z. Ni, and B. Dai. Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians, 2024, 2403.17898.

[27] Saswat Mallick and Rahul Goel, B. Kerbl, F. Vicente Carrasco, M. Steinberger, and F. De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, 2024.

[28] Y. Seo, Y. S. Choi, H. S. Son, and Y. Uh. Flod: Integrating flexible level of detail into 3d gaussian splatting for customizable rendering, 2024, 2408.12894.

[29] X. Sun, J. C. Lee, D. Rho, J. H. Ko, U. Ali, and E. Park. F-3dgs: Factorized coordinates and representations for 3d gaussian splatting, 2024, 2405.17083.

[30] H. Wang, H. Zhu, T. He, R. Feng, J. Deng, J. Bian, and Z. Chen. End-to-end rate-distortion optimized 3d gaussian representation, 2024, 2406.01597.

[31] M. Wu and T. Tuytelaars. Implicit gaussian splatting with efficient multi-level tri-plane representation. *arXiv preprint arXiv:2408.10041*, 2024.

[32] T. Wu, Y.-J. Yuan, L.-X. Zhang, J. Yang, Y.-P. Cao, L.-Q. Yan, and L. Gao. Recent advances in 3d gaussian splatting. *Computational Visual Media*, pages 1–30, 2024.

[33] S. Xie, W. Zhang, C. Tang, Y. Bai, R. Lu, S. Ge, and Z. Wang. Mesongs: Post-training compression of 3d gaussians via efficient attribute transformation. In *European Conference on Computer Vision*. Springer, 2024.

[34] Z. Yan, W. F. Low, Y. Chen, and G. H. Lee. Multi-scale 3d gaussian splatting for anti-aliased rendering, 2024, 2311.17089.

[35] R. Yang, Z. Zhu, Z. Jiang, B. Ye, X. Chen, Y. Zhang, Y. Chen, J. Zhao, and H. Zhao. Spectrally pruned gaussian fields with neural compensation, 2024, 2405.00676.

[36] J. Zhang, F. Zhan, M. Xu, S. Lu, and E. Xing. Fregs: 3d gaussian splatting with progressive frequency regularization, 2024, 2403.06908.

[37] Z. Zhang, W. Hu, Y. Lao, T. He, and H. Zhao. Pixel-gs: Density control with pixel-aware gradient for 3d gaussian splatting, 2024, 2403.15530.