5/1/2025, 6:14:21 PM_3D Gaussian Splatting Techniques

0. 3D Gaussian Splatting Techniques

1. Introduction

Three-dimensional (3D) reconstruction and novel view synthesis (NVS) are fundamental problems in computer vision and graphics, aiming to infer the 3D structure of scenes or objects from a set of 2D images and generate photorealistic images from arbitrary viewpoints [3,7,31,38]. Traditional methods often rely on explicit geometric representations like meshes or point clouds or implicit volumetric representations, facing challenges such as slow rendering speed, high computational costs, and difficulties in handling complex scenes or achieving photorealistic quality [1,5,18,37].

Neural Radiance Fields (NeRF) emerged as a revolutionary implicit representation technique that models a scene’ s radiance field using a neural network [5,10,20,21,28,35,37]. While NeRF achieved unprecedented fidelity in NVS, it typically suffers from slow training times and, critically, requires computationally expensive volume rendering for each new view, hindering real-time applications—especially at high resolutions [2,5,7,10,11,15,17,28,29,30,33,36,37].

In response to these limitations, 3D Gaussian Splatting (3DGS) was introduced in 2023 as a novel approach that achieves state-of-the-art visual quality in NVS with significantly improved rendering efficiency, enabling real-time performance [1,2,4,10,14,15,16,17,18,20,21,22,24,25,28,29,30,31,33,35,36,37]. Unlike NeRF’ s implicit neural representation, 3DGS employs an explicit scene representation consisting of a set of 3D Gaussian ellipsoids [9,23,25,29,33]. Each Gaussian is defined by its spatial location ∈ ⊠ (denoted by μ), covariance matrix (Σ), color c, and opacity (α) [14]. The key to its efficiency lies in a differentiable rendering pipeline based on rasterization, which projects the 3D Gaussians onto the 2D image plane as elliptical splats and employs alpha blending to determine the final pixel color [14,18,26,29,33]. The method typically initializes from a sparse point cloud obtained via Structure-from-Motion (SfM) and optimizes the Gaussian parameters through interleaved optimization and density control [7,16,17,21,23,36,37].

Feature	NeRF (Limitations addressed by 3DGS)	3D Gaussian Splatting (Advantages)
Rendering Speed	Slow (Volume Rendering)	Faster (Real-time, Rasterization-based)
Representation	Implicit (Neural Network)	Explicit (3D Gaussian Primitives)
Fidelity	High	State-of-the-art (Detailed, HDR capable)
Editability	Challenging	Easier (Direct manipulation of primitives)
Training Times	Often Slow	Competitive or Superior
Rendering Method	Ray Casting + Volume Rendering	Rasterization + Alpha Blending (Splatting)
Initialization	Often random or based on sparse	Typically from SfM Point Cloud

3DGS offers several key advantages that have led to its rapid adoption and the growing research interest since its introduction [5,14,17,22,27,30,35]:

· **Faster Rendering Speed:** A major breakthrough is the ability to render scenes in real time (often $\geqslant$30 fps) at high resolutions like 1080p, a significant improvement over NeRF [7,14,15,16,17,18,20,25,26,30]. Specific implementations have demonstrated speeds exceeding 100 FPS or even 1000× speedups over related volumetric methods [11,15].

- **Explicit Representation:** Representing a scene as a collection of discrete 3D Gaussians offers a more intuitive and interpretable structure compared to implicit neural networks [9,23,25,29,33]. This explicit nature also avoids the computational overhead that comes with rendering large volumes of empty space in volumetric methods [5,23].
- **High Fidelity:** 3DGS achieves state-of-the-art visual quality in novel view synthesis, producing detailed and realistic renderings [1,4,10,14,16,17,18,20,28,30,31,34]. It can reproduce high-quality texture details and manage challenging scenarios, including High Dynamic Range (HDR) scenes [11,15].
- **Editability:** The explicit structure facilitates various editing tasks—ranging from geometric manipulation and appearance modification to dynamic scene reconstruction and physical simulation—which are often more challenging with implicit representations [25,29,32,33].
- **Competitive Training Times:** Despite achieving high quality and speed, 3DGS maintains training times that are competitive with or even superior to previous methods such as NeRF [10,12,16,17,24,28,30,31].

While 3DGS presents significant advancements, it also faces challenges such as limitations in accurately reconstructing geometry, handling dynamic scenes, achieving artistic control, high data requirements, and potential vulnerabilities related to data manipulation that can impact memory usage [9,12,18]. Ongoing research explores extensions and applications of 3DGS in areas including dynamic scenes [8,32], large-scale environments [12], segmentation [6], semantic understanding, SLAM, digital humans, virtual/augmented reality, and simulations [2,4,21,29,30,33,34].

This survey provides a comprehensive overview of 3D Gaussian Splatting techniques. It delves into the underlying principles, including scene representation and the differentiable rendering pipeline, and analyzes various optimization strategies and density control mechanisms. Furthermore, the discussion covers the wide range of applications enabled by 3DGS, recent extensions, and advancements in the field, while also highlighting current technical challenges and potential future research directions [2,24,25].

2. Background and Related Work

Representation Type	Examples	Structure	Advantages	Challenges	Rendering Compatibility
Explicit	Meshes, Point Clouds, 3D Gaussians	Direct geometric elements (vertices, faces, points, ellipsoids)	Intuitive, Interpretable, GPU Rasterization Comp.	Complex for intricate scenes, Memory for dense, Redundancy	High (GPU/CUDA Rasterization)
Implicit	NeRF, SDFs	Neural Network mapping 3D point to property	Compact memory, High fidelity (view-dependent)	Computationally Intensive Queries, Slow Rendering	Low (Network Query per sample)

The accurate reconstruction and realistic rendering of three-dimensional scenes from collections of two-dimensional images have long been central challenges in computer graphics and computer vision. Traditionally, scene reconstruction has relied on geometric methods such as Structure from Motion (SfM) and Multi-View Stereo (MVS) [29,33,37]. These techniques estimate camera poses and generate sparse or dense 3D point clouds or meshes by analyzing correspondences across multiple views [4,7,35]. While effective for generating foundational 3D data and compatible with established rendering pipelines, traditional methods can struggle with reconstruction quality in challenging areas and may require significant memory for dense representations or storing input images [12,37].

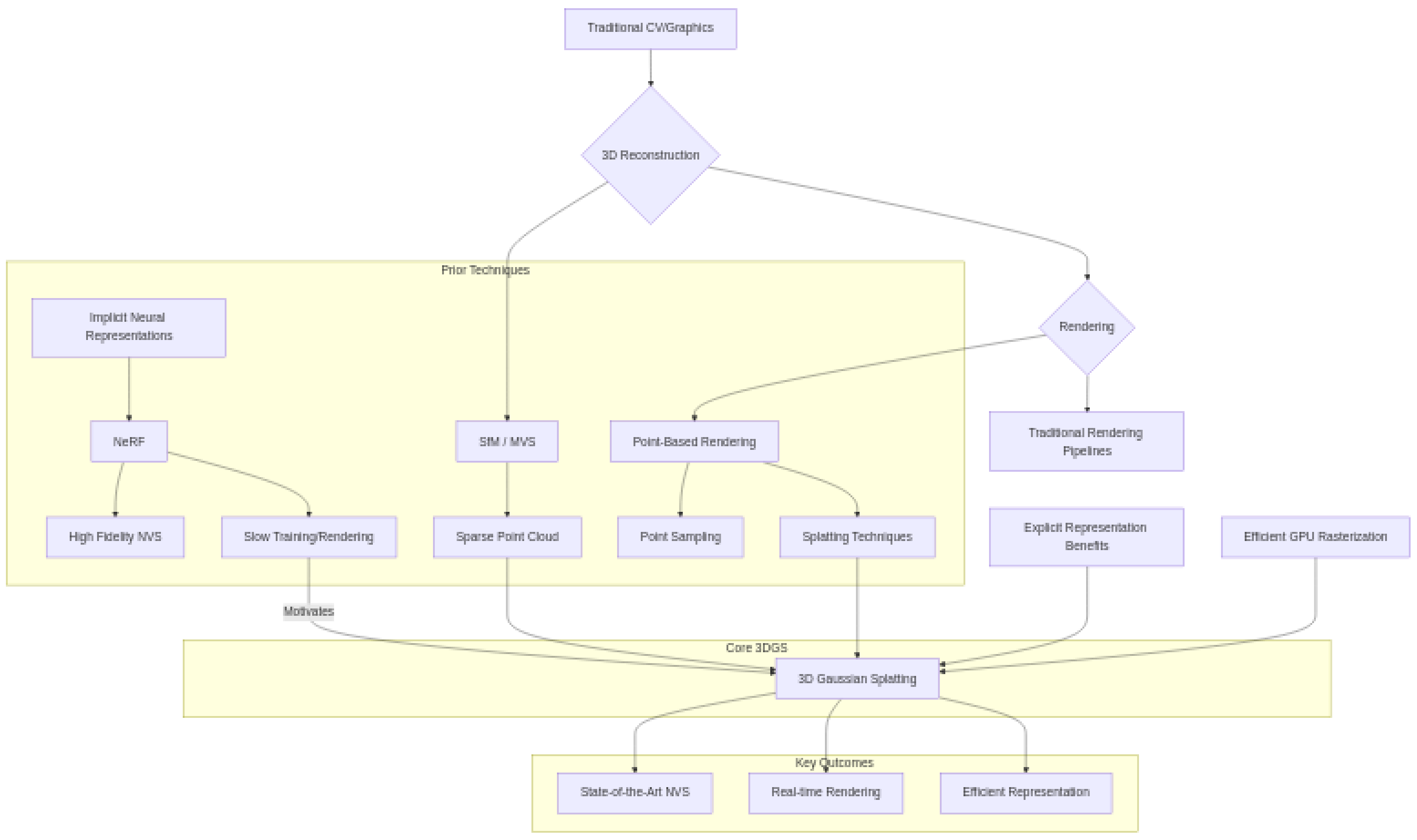
The landscape of geometric representations has evolved significantly, transitioning from traditional explicit methods to neural implicit representations and, more recently, hybrid approaches [5,8]. Explicit representations, such as polygon meshes and point clouds, store geometric elements directly [21]. Meshes use interconnected vertices, edges, and faces, enabling compatibility with modeling tools but facing complexity and memory issues for intricate scenes [3,12]. Point clouds, composed of discrete, independent points, simplify some operations and are efficient for real-time rasterization on GPUs [3,31], though they can suffer from data redundancy and difficulties representing smooth surfaces or empty space [3,36]. Implicit methods, on the other hand, use neural networks to represent geometry and appearance indirectly, querying properties like color and density at arbitrary 3D points [6,21]. This can lead to compact representations and high-fidelity reconstruction [3], but querying the network can be computationally intensive [21].

Neural Radiance Fields (NeRF), introduced in 2020, represents a major advancement in implicit neural rendering [10,17,28,30,35]. NeRF models a scene's radiance field using a neural network, typically an MLP, that maps 5D inputs (3D position and 2D viewing direction) to RGB color and volume density [3,5,7,9,25]. Novel views are synthesized by casting rays through pixels and using volume rendering to integrate color and density sampled along each ray, yielding high-quality visual output [1,3,5,7,18,20].

Despite its success in generating photorealistic novel views, NeRF is characterized by significant computational bottlenecks. Training NeRF models is often slow [11,16,30,36], and rendering is particularly computationally expensive due to the need for dense sampling along rays and repeated neural network queries for each pixel, hindering real-time performance [2,5,7,11,20,30,36]. For instance, rendering a modest 400x400 image can take seconds for some NeRF variants [11]. NeRF also struggles with representing empty space efficiently and its performance can be sensitive to scene characteristics [36]. These limitations motivated the development of acceleration techniques, such as using spatial data structures or different encodings (e.g., NGP, Plenoxels), and spurred the search for alternative rendering paradigms that could achieve real-time performance while maintaining high quality [11,20,32,37].

Point-based rendering offers an alternative paradigm that avoids the complexities of mesh structures and the sampling burden of implicit volumetric methods [5,27,37]. Representing scenes as discrete points is efficient for real-time rasterization [2,18]. Traditional point-based rendering includes point sampling and splatting [29,33]. Simple point sampling can result in rendering artifacts like holes and aliasing [27,36,37]. Splatting techniques address this by projecting point primitives (e.g., ellipses) onto the image plane to cover areas larger than single pixels, effectively diffusing the influence of 3D points and creating a more continuous appearance [20,36,37]. This forward mapping approach is fundamentally different from ray tracing [1,22].

3D Gaussian Splatting (3DGS) builds upon these prior techniques, particularly point-based rendering and splatting, while addressing the efficiency limitations of NeRF [5,20,29,33,37]. Instead of discrete points or a neural network, 3DGS represents a scene using a set of learnable, anisotropic 3D Gaussian distributions initialized from sparse point clouds generated by traditional SfM methods [11,15,18,20]. Crucially, this Gaussian representation is explicit [6,21], enabling efficient, high-quality rendering via differentiable rasterization on GPUs [2,31].



By leveraging an explicit, splattable primitive, 3DGS achieves significantly faster training and rendering speeds compared to NeRF's volume rendering, while using adaptive Gaussians to efficiently represent scene complexity [9,29,33].

2.1 Geometric Representations

The choice of geometric representation is fundamental to 3D scene reconstruction, rendering, and downstream applications. The field has seen an evolution through various methods, ranging from traditional explicit structures to modern implicit and hybrid approaches [8]. Understanding these methods provides context for the design choices in techniques like 3D Gaussian Splatting.

Traditional explicit representations, such as polygon meshes and point clouds, store geometry directly. Mesh-based models define surfaces using interconnected vertices, edges, and faces, often triangular patches [3,18]. This structure is highly compatible with 3D modeling software [3] and enables direct access to geometric elements [21]. However, their inherent dependencies between elements can lead to computational complexity, especially for intricate geometries. Representing complex scenes accurately can also require a large number of polygons, impacting memory efficiency [3,12]. Point clouds, on the other hand, represent a scene as a collection of discrete, independent points, each possessing attributes like position, color, and opacity [3,18]. The independence of points eliminates complex dependencies, simplifying certain operations [3]. Both meshes and point clouds benefit from being explicit representations, which makes them highly compatible with GPU/CUDA-based rasterization pipelines, facilitating rapid rendering [31]. Despite their advantages in rendering speed, point clouds can suffer from data redundancy, particularly when representing smooth surfaces or empty regions, potentially leading to large data sizes [3].

In contrast, neural implicit surfaces, exemplified by methods like Neural Radiance Fields (NeRF), represent geometry implicitly [6,21]. These models typically use a neural network, such as a multilayer perceptron (MLP), to infer properties like color and density at any given 3D coordinate and viewing direction [3,21]. Implicit representations avoid explicitly storing geometry, often resulting in a compact memory footprint [3] and enabling high-fidelity reconstruction, particularly for complex visual effects like view-dependent appearances. However, they infer point properties indirectly [21], and query operations often require network inference, which can be computationally intensive and impact real-time performance compared to explicit rasterization methods. Furthermore, accurately reconstructing complex or large-scale scenes with implicit methods presents challenges in balancing model capacity, geometric accuracy, and memory efficiency [12].

3D Gaussian Splatting adopts a distinct geometric representation: a set of learnable 3D Gaussian distributions [15,18,20]. This representation is initialized from sparse point clouds, typically obtained via Structure from Motion (SfM) [15,20]. Unlike discrete points or mesh facets, each Gaussian is a continuous, anisotropic ellipsoid, allowing for a more detailed and view-dependent scene representation [20]. Crucially, the 3D Gaussian representation is explicit, similar to point clouds and meshes [6,21]. This explicit nature enables compatibility with efficient GPU-based rasterization pipelines [31] and facilitates fast optimization and efficient operations like segmentation [6]. By using Gaussians, 3DGS retains some benefits associated with continuous volumetric fields, such as handling transparency and view-dependent effects, while avoiding the computational burden of sampling empty space, a common challenge in traditional volumetric or some implicit methods [20]. Thus, 3DGS leverages the advantages of explicit representations for efficient processing and rendering while employing a richer, learnable primitive (the anisotropic Gaussian) to capture scene complexity and detail.

2.2 Neural Radiance Fields (NeRF) and Implicit Methods

Neural Radiance Fields (NeRF), introduced in 2020, significantly advanced the field of novel view synthesis from sets of input images and their corresponding camera poses [10,17,28,30]. At its core, NeRF employs a neural network to implicitly model a 3D scene's radiance field [5,9,25]. This neural network, typically a multi-layer perceptron (MLP), takes a 5D input consisting of a 3D spatial coordinate (x, y, z) and a 2D viewing direction (θ, ϕ) [7,35] and outputs the color (RGB) and volume density (σ) at that point and view direction [3,7,19,27]. This implicit representation allows for a continuous scene representation [18,31].

Image generation in NeRF is achieved through passive ray tracing and volume rendering [1,5]. For each pixel in a target novel view, a ray is cast into the scene [1,21]. Multiple points are sampled along this ray, and the neural network is queried at each sampled point to obtain its predicted color and volume density [1,2,7,11]. These sampled values are then integrated along the ray using volume rendering techniques to determine the final pixel color [1,5,7].

NeRF offers several advantages, including the ability to produce high-quality visual representations [3,16,18,20] and a potentially small memory footprint due to its implicit nature [3]. Its continuous representation is also beneficial for optimization [31].

Despite its success and quality, NeRF is characterized by significant limitations, particularly concerning computational efficiency. Training NeRF models is often slow [11,16,30,36], and rendering novel views is computationally intensive and time-consuming [2,5,7,11,20,30,32,36]. The requirement to trace rays and sample numerous points along each ray, followed by querying the neural network for each sample, makes volume rendering computationally expensive and hinders real-time performance, especially for high resolutions like 1080p or unconstrained scenes [2,7,11,20]. For instance, HDR-NeRF, an advanced NeRF variant, reportedly requires 9 hours for training and 8.2 seconds to render a single 400x400 image [11]. The stochastic sampling during rendering can also introduce costly noise [31]. Furthermore, NeRF struggles with efficiently representing empty spaces [36] and its performance can be heavily dependent on the scene characteristics and image capture methodology [36]. These significant bottlenecks in training and rendering speed, along with challenges in achieving consistent quality and handling specific tasks like segmentation or large-scale deformation [6,8], were key drivers for the development of alternative techniques like 3D Gaussian Splatting [11,20,32], which aims to overcome these limitations, particularly achieving faster rendering [29,33].

Beyond basic NeRF, other neural implicit methods and acceleration techniques have been explored to address some of these issues. These include methods using spatial data structures or different encodings to accelerate volumetric ray-marching, such as NGP and Plenoxels [36]. MERF, for example, attempts to accelerate rendering by pre-baking ray tracing results into volume textures, though this can compromise detail and shape fidelity [37]. Other neural implicit representations have also been applied to tasks like 3D scene reconstruction, facing similar computational challenges [6].

2.3 Point-Based Rendering and Traditional Techniques

Point-based rendering represents and visualizes 3D scenes using discrete geometric points rather than traditional polygon meshes [5,27]. This approach is particularly well-suited for rendering complex, unstructured, or sparse geometric data, such as point clouds obtained from 3D scanning or reconstruction processes [2,5,27,37]. Discrete point cloud representations are noted for their efficiency and precision in recording only occupied space and facilitating real-time rendering through rasterization on modern GPUs, differing from methods reliant on sampling [2]. Point-based methods can enable real-time rendering and are often easier to edit and adapt for dynamic scenes compared to other representations [18].

Traditionally, point-based rendering can be broadly categorized into point sampling and splatting techniques [29,33]. Point sampling involves rasterizing fixed-size, unstructured point sets using graphics APIs or parallel software rasterization [36,37]. However, this method commonly suffers from rendering artifacts such as holes, aliasing, and discontinuity, particularly when the projected point size is small or point density is insufficient [27,36,37]. Splatting addresses these limitations by projecting point primitives, such as circles, ellipses, or ellipsoids, to cover an area larger than a single pixel [36,37]. This technique creates a diffusion-like effect on the image plane when projecting 3D points, and the combined effect forms the final image [20]. Splatting is a forward mapping or rasterization process, treating each 3D element (like a voxel or Gaussian) as a "fuzzy sphere" or "luminous particle" whose influence on individual pixels is computed [1,21,22]. This contrasts with passive ray tracing methods which trace rays from the camera into the scene [1,21].

3D Gaussian Splatting (3DGS) leverages and extends the traditional splatting technique for rendering point clouds [20]. It utilizes anisotropic 3D Gaussian functions to represent the scene, effectively addressing issues like holes and aliasing often encountered in traditional point-based rendering by providing a more coherent and continuous scene representation [27]. These Gaussians, represented as ellipsoids, are bound to attributes such as opacity, color, and spherical harmonic coefficients for alpha blending during rendering [12]. 3DGS employs rasterization for efficient and high-quality rendering, combining aspects of explicit representation (the Gaussians) and implicit features (learnable attributes) [8,15]. By using splatting, 3DGS maximizes the utility of initial point cloud data [15].

Before rendering techniques like 3DGS can synthesize novel views, an initial 3D representation of the scene is required. Traditional computer vision techniques, such as Structure from Motion (SfM) and Multi-View Stereo (MVS), play a crucial role in generating this foundational 3D data [29,33,37]. SfM analyzes a collection of images taken from different viewpoints to simultaneously recover the camera poses and a sparse set of 3D points corresponding to features observed across multiple images [4,7,35]. Libraries like COLMAP are commonly used for this purpose [7]. MVS techniques build upon the camera poses and sparse points provided by SfM to generate a denser 3D reconstruction, typically in the form of a dense point cloud or mesh. These traditional methods provide the initial geometric structure from which subsequent representations are derived. Specifically, 3DGS commonly initializes its set of 3D Gaussians directly from the sparse point cloud generated by SfM [4,7,11,20]. This initialization provides a starting point for the optimization process, although it is noted that some point-based rendering methods relying heavily on MVS initialization may inherit artifacts from under- or over-reconstructed regions [36,37].

3. 3D Gaussian Splatting: Core Methodology

Parameter	Description	Representation / How Defined	Function in 3DGS Scene Representation
Spatial Location	Center of the Gaussian in 3D space	3D vector (μ or (x, y, z))	Defines position of the primitive
Covariance	Shape and Orientation of the Gaussian (anisotropy)	3×3 matrix (Σ) / Scaling (s) & Rotation (q)	Controls stretching and orientation
Color	Appearance	RGB values or Spherical Harmonic (SH) coefficients	Models surface appearance (view-dependent with SH)
Opacity	Transparency	Single scalar value (α)	Controls visibility and blending

The 3D Gaussian Splatting (3DGS) methodology provides an explicit, high-fidelity representation for 3D scenes, enabling real-time rendering and efficient optimization [5,7,16,19,36]. Unlike implicit neural representations that rely on complex network evaluations during rendering, 3DGS represents a scene as a collection of numerous 3D Gaussian distributions, often referred to as "Gaussian ellipsoids" or "splats" [1,4,14,25]. Each Gaussian is a volumetric primitive characterized by a set of learnable parameters that define its spatial properties and appearance [3,5,14,19,21,37].

The core parameters defining each 3D Gaussian are its spatial location, covariance, color, and opacity [14,23,31]. The spatial location is given by the mean vector ⍰ in 3D space [4,20]. The shape and orientation are captured by an anisotropic covariance matrix Σ, which can be parameterized by a scaling vector s and a rotation quaternion q [16,21]. Color information is typically stored using spherical harmonic (SH) coefficients to model view-dependent appearance effects, although simple

RGB values can also be used [1,3,20]. Opacity, denoted by ⌧, controls the transparency of the Gaussian [1,14,20]. Mathematically, a 3D Gaussian distribution is defined by its mean ⌧ and covariance Σ [24]. These parameters are initialized, often from a sparse point cloud obtained via Structure from Motion (SfM) [20,22,33], and collectively optimized during the training process [26,27].

The rendering pipeline is central to 3DGS's efficiency and differentiability [5,7,19,29,33,37]. It operates by projecting the 3D Gaussians onto the 2D image plane, a process called "splatting" [1,14,23,27]. The 3D covariance matrix Σ is transformed into a 2D covariance matrix Σ′ on the image plane based on the camera parameters and perspective projection, typically formulated as

Σ′ = ⌧ ⌧ Σ ⌧⌧ ⌧⌧

where ⌧ is the view transformation and ⌧ is the Jacobian of the affine approximation of the projective transformation [3,7,14]. To correctly composite the projected Gaussians, they are sorted in back-to-front order relative to the camera viewpoint based on their depth [22,27]. Pixel colors are then computed by alpha blending the contributions of the sorted, overlapping Gaussians [1,2,14]. The final color C at a pixel is determined by accumulating the contributions c⌧ and opacities ⌧⌧′ of N Gaussians sorted by depth (i = 1 ⋯ N) via the formula:

C = Σ₍⌧⌧⌧,⌧ c⌧ ⌧⌧′ ∏ ⌧⌧⌧⌧⌧⌧⌧⌧¹⌧ (1 − ⌧⌧′)

[19,20]. This rendering process is implemented using an efficient tile-based rasterizer on the GPU, which parallelizes the projection, sorting, and blending operations across image tiles, enabling high-resolution, real-time rendering performance [7,15,27,29,33,36,37].

The core optimization process in 3DGS iteratively refines the parameters of the 3D Gaussians [4,5,27]. This involves minimizing a loss function by comparing the rendered images to the ground truth input images using gradient descent [3,4,19,22]. The loss function typically combines a photometric term, such as the L_1 norm [7,20], with a structural similarity (SSIM) term or its variants like D-SSIM [7,19,26] to improve perceptual quality. A common form of the total loss is a weighted sum:

⌧ = (1 − λ) ⌧⌧ + λ ⌧⌧SSIM⁄D⌧SSIM⌧

[19,20]. The gradients of this loss with respect to the Gaussian parameters (position, covariance, color, opacity) are computed via backpropagation through the differentiable rendering pipeline, enabling updates using optimizers like SGD or Adam [7,15,20].

Crucially, 3DGS employs adaptive density control mechanisms interleaved with parameter optimization [7,15,26,29,33,37]. This process dynamically adjusts the number and distribution of Gaussians to efficiently represent scene complexity [4,17,30]. Densification adds Gaussians to under-reconstructed areas or regions with high gradients, typically through cloning existing Gaussians or splitting large ones into smaller ones [2,20,22]. Pruning removes redundant or less effective Gaussians, primarily those with low opacity or that are overly large [2,7,20]. This adaptive density control is vital for achieving a compact yet detailed scene representation and managing computational resources [9,27]. The interplay between differentiable rendering, parameter optimization, and adaptive density control forms the core of the 3DGS pipeline, enabling rapid and high-quality novel view synthesis from multi-view images.

3.1 3D Gaussian Representation and Initialization

Three-dimensional Gaussian Splatting (3DGS) represents a scene as a collection of numerous 3D Gaussian distributions, often referred to as "Gaussian ellipsoids" [1,4,18,22]. Each individual 3D Gaussian is parameterized by a set of attributes that collectively define its spatial presence and appearance [3,5,14,19,21,37].

The core parameters characterizing each 3D Gaussian include its spatial location, covariance, color, and opacity. The spatial location, represented by a 3D vector μ (mean) or coordinates (x, y, z), defines the center of the Gaussian in 3D space [3,4,14,20,25,27]. The shape and orientation of the Gaussian are determined by a 3×3 covariance matrix Σ [1,3,5,14,20,21,27,37]. This matrix implicitly encodes rotation and scaling factors [21], which can be optimized via parameters like scale s and quaternion q through gradient descent [21]. Color information can be stored directly as RGB values or, more commonly, as spherical harmonic (SH) coefficients [1,3,5,20,27,31]. Opacity, denoted by α, is a single value representing the transparency of the Gaussian [2,3,5,7,12,14,20,27,31,37]. Together with position, scaling, and rotation, these parameters allow each Gaussian to have a clear physical significance [13].

These parameters are fundamental to modeling the scene's appearance and geometry. The position parameter dictates the location of the Gaussian primitive in 3D space [1,3,4,20]. The anisotropic nature of the 3D Gaussians, defined by the covariance matrix, allows them to stretch and orient arbitrarily in space [20,21]. This anisotropy is crucial for representing complex shapes and capturing fine geometric details accurately by dictating how the splat stretches or scales [4]. Color and spherical harmonic coefficients model the surface appearance and texture [3]. Notably, spherical harmonics enable the representation of view-dependent color effects, which is essential for photorealistic rendering from various viewpoints [2,3,4,20,27]. Using multiple SH coefficients allows capturing more detailed color variations [3]. Opacity controls the visibility of each Gaussian and is critical for correctly blending multiple splats during the rendering process to form a coherent scene [1,4,12,36]. All these attributes are learnable parameters, optimized through backpropagation during the training process to match the known dataset images [27]. The model intelligently adapts resources based on scene complexity, preserving continuous volumetric attributes while avoiding computation in empty areas [9,30].

The initialization of the 3DGS scene is a critical first step. The most common method involves starting from a sparse point cloud, typically obtained through classical Structure from Motion (SfM) pipelines like COLMAP [1,2,7,11,16,17,19,20,22,26,27,33]. SfM analyzes overlapping features across multiple 2D input images to estimate the 3D locations of points and the camera parameters [11,22]. Each point in this initial sparse cloud is then converted into a 3D Gaussian primitive [4,19,20]. The center of the Gaussian is typically set to the 3D position of the corresponding point cloud point, while other parameters, such as covariance (scale and rotation) and opacity, are often initialized randomly or based on simple heuristics [19]. While SfM-based initialization from sparse points is prevalent, particularly after processes involving steps like COLMAP estimation [26], other initialization strategies exist, including random initialization [2,27], voxel grid sampling, or dense initialization using depth maps [14]. Initialization from sparse points is noted for preserving continuous volumetric properties efficiently [30]. Once initialized, the collection of millions (typically 0.5 to 5 million) of these 3D Gaussians forms the initial representation of the scene [27,37], which is then refined through an optimization process.

3.2 Differentiable Rendering Pipeline

The core of 3D Gaussian Splatting (3DGS) lies in its differentiable rendering pipeline, which facilitates efficient novel view synthesis and optimization [5,7,19,29,33,37]. This pipeline involves projecting 3D Gaussian primitives onto the 2D image plane, calculating their contributions to pixel colors, and blending these contributions to produce the final rendered image [1,2]. Unlike methods based on ray marching, such as NeRF, which process each pixel by sampling along a ray, 3DGS projects explicit geometric primitives (Gaussians) onto the image, a process termed “splatting” [5,18,23].

The initial step of the rendering process is the projection of the 3D Gaussians from world space to the 2D image plane [11,21,22]. This requires transforming the 3D Gaussian's properties, particularly its mean (position) and covariance matrix, based on camera parameters (intrinsics and extrinsics) and the perspective projection [3,19,20]. The 3D covariance matrix Σ is transformed into a 2D covariance matrix Σ′ on the image plane. This transformation involves the view transformation matrix W and the Jacobian J of the affine approximation of the projective transformation [7,14]. The projected 2D covariance Σ′ can be computed by applying these transformations to the 3D covariance Σ, typically represented as

$$\Sigma_{2D} = J\Sigma J^T$$

or

$$\Sigma' = \boldsymbol{J}\,\boldsymbol{W}\,\Sigma\,\boldsymbol{W}^\top\,\boldsymbol{J}^\top.$$

After projection, the contributions of overlapping Gaussians at each pixel are calculated and combined. A crucial aspect for achieving correct transparency and view-dependent effects is the sorting of projected Gaussians [27]. Gaussians must be rendered in back-to-front order relative to the camera view to correctly apply alpha blending [22]. This involves sorting the projected 2D Gaussians based on their depth from the camera [5,11,20].

Pixel colors are determined by accumulating the contributions of all Gaussians whose 2D projections overlap the pixel location using alpha blending [1,2,6,14]. Alpha blending is a standard technique for mixing colors based on transparency values [1]. The opacity

$$\alpha_i' = \alpha_i \times \exp\left(-\frac{1}{2}\,(x' - \boldsymbol{\mu}_i')^\top\,(\boldsymbol{\Sigma}_i')^{-1}\,(x' - \boldsymbol{\mu}_i')\right)$$

of a Gaussian i at a projected point x′ on the image plane is calculated based on its 3D opacity α_i and its 2D Gaussian distribution evaluated at x′ [20].

The final color C of a pixel is computed by compositing the colors c_i and opacities α_i′ of the sorted overlapping Gaussians ☒ using the alpha blending formula

$$C = \sum_{i \in \mathcal{N}} c_i \, \alpha_i' \prod_{j=1}^{i-1} \left(1 - \alpha_j'\right) .$$

This process is deterministic and differentiable [19].

A key factor in 3DGS's ability to achieve real-time rendering speeds is its efficient rasterization process, particularly the use of tile-based approaches [7,15,29,33,36,37]. Instead of processing pixels individually, the image plane is divided into non-overlapping regions or tiles, typically 16×16 pixels [20,27]. Gaussians are assigned to the tiles they overlap [11,37]. If a Gaussian intersects multiple tiles, it is replicated and assigned to each relevant tile with a tile identifier [7,27]. Within each tile, visible Gaussians are selected [20], sorted by depth (often using a combination of tile ID and depth for efficient parallel processing) [27], and then splatted and blended in parallel using GPU capabilities like CUDA blocks and threads [15,20,23,26,27]. This tile-based rasterization enables high-resolution, high frame rate rendering by effectively distributing the computational load across GPU cores [2,31]. The rendering algorithm also supports anisotropic splatting, further contributing to accelerated rendering and training [16,17,30].

The differentiability of the entire rendering pipeline is a significant advantage [25]. It allows for the calculation of gradients with respect to the 3D Gaussian properties (mean, covariance, color, opacity) based on the difference between the rendered image and the ground truth image. These gradients can then be used in standard gradient-based optimization techniques, such as stochastic gradient descent, to refine the parameters of the 3D Gaussians during the training phase [23]. This enables the model to accurately reconstruct the 3D scene and synthesize high-quality novel views by minimizing a loss function based on the rendered output, for instance, the squared error between a rendered mask and a ground truth mask

$$\mathcal{F} = \sum_{i \in \text{pixels}} \left(M_i - \hat{M}_i\right)^2 ,$$

as described in [6]. The rapid backpropagation capability facilitated by the differentiable GPU rendering is crucial for accelerating the training process [23].

3.3 Core Optimization Process

The core of the 3D Gaussian Splatting (3DGS) framework lies in its optimization process, which iteratively refines a set of 3D Gaussian primitives to accurately represent a scene [4,5,24,27]. This process involves two primary aspects: parameter optimization and adaptive density control, which are alternated and interleaved with differentiable rendering [5,16,24,27,30].

Parameter optimization focuses on adjusting the attributes of each Gaussian, including its position, opacity (α), anisotropic covariance (defined by scaling and rotation), and spherical harmonic (SH) coefficients representing color and view-dependent effects [17,19,27,30]. Initial parameters are often derived from a preliminary Structure from Motion (SfM) point cloud, where each point is converted into a Gaussian [1,3]. The refinement is driven by minimizing a loss function that quantifies the difference between the rendered image and the ground truth image from input views [4,19,22].

The loss function typically comprises photometric errors, predominantly utilizing a combination of L1 loss and a structural similarity index [1,7,19,20,26]. The L1 loss measures the average absolute difference between corresponding pixel values in the rendered ($\hat{I}$) and ground truth (I) images [3,19]:

$$\mathcal{L}_1 = \frac{1}{N} \sum_{i=1}^{N} |I_i - \hat{I}_i|$$

[3,19]

where N is the number of pixels. Structural Similarity (SSIM), Depth-aware SSIM (D-SSIM), or variations like LD-SSIM are incorporated to capture perceptual differences related to luminance, contrast, and structure, often providing results that are more visually pleasing than L1 or L2 losses alone [7,19,26]. The total loss is commonly a weighted sum of L1 and a structural similarity term [7,11,19,20], such as:

$\mathcal{L} = (1-\lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{SSIM/D-SSIM}}$

[7,19,20]

where λ is a weighting factor. D-SSIM specifically enhances the structural similarity measure with depth information [7]. Specialized loss functions may also be used for specific applications, such as combined LDR and HDR losses for High Dynamic Range scenes [11].

The minimization of this loss function and the subsequent parameter updates are performed using gradient descent-based optimization algorithms [3]. Stochastic Gradient Descent (SGD) is a common choice [1,20,22,23], while the Adam optimizer is also explicitly utilized [7]. Backpropagation is used to calculate gradients and update the parameters of all Gaussians contributing to the rendered image [3,15]. Activation functions are applied to constrain certain parameters, such as a sigmoid for opacity to keep it within and an exponential function for scale [20].

Adaptive density control is crucial because the initial number and distribution of Gaussians may be insufficient to represent the scene accurately [2,4,17,30]. This mechanism dynamically adjusts the number of Gaussians throughout the optimization process through densification (adding Gaussians) and pruning (removing Gaussians) [4,15,17,24,26,27,30].

Densification strategies are guided by the magnitude of gradients, particularly the position gradient in view space, which indicates areas that are under-reconstructed or where the representation needs further refinement [1,2,20,23,26]. This process focuses on areas with missing geometry or overly dispersed Gaussians [27]. The primary methods are cloning and splitting:

- **Cloning**: Gaussians are duplicated, often when their position gradient is large, and the copy may be moved slightly in the gradient direction [1,2,20,22,23,26]. This is typically applied to handle under-reconstructed regions or small Gaussians requiring more density [1,20,22,23].
- **Splitting**: A single Gaussian is divided into multiple smaller ones, commonly applied to large Gaussians or those in over-reconstructed areas with high variance or large gradients [1,2,20,22,23,26]. Splitting may divide the scale by a fixed factor (e.g., scale divided by 16 or a ratio of 1.6) [7,26]. Some methods also incorporate regularization terms to guide the splitting process based on local geometry [8].

Pruning removes Gaussians that are redundant or have minimal impact on the rendered image [2]. This primarily targets Gaussians with opacity below a specified threshold, making them nearly transparent [1,7,20,23,27]. Overly large Gaussians may also be pruned [2]. To prevent the accumulation of unnecessary Gaussians or "floaters" in space, the opacity of all Gaussians is periodically reset to near zero [2,7,20].

These optimization steps—parameter updates via gradient descent and adaptive density control—are tightly interleaved with the differentiable rendering process [5,16,27,30]. In each iteration, the current set of 3D Gaussians is rendered from a selected viewpoint using differentiable Gaussian rasterization [1,23]. The rendered image is then compared to the corresponding ground truth image to calculate the loss [1,3,22,23]. The gradients of the loss with respect to the Gaussian parameters are computed via backpropagation, driving the parameter updates and informing the density control decisions [1,3,15,23]. This cycle of rendering, loss calculation, parameter update, and density adjustment enables the 3D Gaussian representation to progressively conform to the scene texture and geometry captured in the input images [4,5,27].

4. Advanced Optimization and Efficiency Techniques

Efficiency Aspect	Challenges	Techniques	Examples / Specifics
Storage	High memory footprint (GBs), Redundancy	Vector Quantization (VQ), Pruning, Alternative Representations	Compact3D (VQ per attribute), EAGLES (VQ), LightGaus (Pruning by importance), CityGaussianV2 (Pruning, Quantization), Scaffold-GS, GES
Training			

	Computationally intensive (Rasterization, Diff), Large scenes	Accelerate atomic operations, Custom CUDA kernels, Optimized Rasterization, Pipeline Optimization	DISTWAR (Atomic Ops), CityGaussianV2 (Unified Pipeline), FlashSplat (Bottleneck Opt)
Rendering	Real-time performance	Tile-based Rasterization, View Frustum Culling, Reduced Gaussian Count, Hardware Acceleration	16x16 Tiles, Sorting by Depth, Alpha Blending, CUDA/GPU, GSCore (Dedicated Hardware)

Efficiency constitutes a fundamental metric for evaluating 3D Gaussian Splatting (3DGS) techniques, encompassing considerations across storage, training, and rendering phases [2]. Significant research efforts are dedicated to optimizing these aspects to enhance the practicality and scalability of 3DGS for diverse applications.

Storage efficiency is a critical challenge due to the substantial memory footprint required by millions of Gaussian primitives used to represent complex scenes [2]. These primitives, storing geometric and appearance attributes, can lead to considerable storage overhead, particularly in large-scale environments [2]. Techniques to mitigate this include attribute compression via Vector Quantization (VQ), exemplified by methods like Compact3D which utilizes distinct codebooks for different attributes, and EAGLES which also employs VQ [2,29,33]. Another strategy involves pruning less important Gaussian primitives, thereby reducing the total primitive count [2]. LightGaus implements pruning based on global importance scores [2], while CityGaussianV2 prunes primitives with low importance scores during the training process itself. The importance score in CityGaussianV2 is calculated based on opacity contribution:

$$Importance = \sum_{i \in \Omega} \sum_{n} o_n$$

where Ω is the set of pixels for the i -th training view, and o_n is the opacity of the n -th primitive [12]. CityGaussianV2 also incorporates quantization for large-scale scene storage [12]. Beyond VQ and pruning, alternative representations like Self-Organizing Gaussians, Scaffold-GS, and GES explore novel ways to enhance representation efficiency [2]. A common trade-off exists wherein higher compression or pruning levels can lead to reduced model size but potentially compromise rendering quality or fidelity.

Training efficiency focuses on accelerating the often computationally intensive process of optimizing Gaussian parameters. Challenges include the demands of rasterization and differentiation steps, as well as managing memory and parallel processing, particularly for large scenes [12]. Techniques like DISTWAR are developed to speed up atomic operations crucial for differentiable rendering within the 3DGS pipeline [2]. The utilization of custom CUDA kernels and optimized fast rasterization further contributes to training acceleration [20]. Pipeline optimizations, such as the unified training and compression process in CityGaussianV2, aim to streamline the workflow and reduce redundant steps [12]. Methods like FlashSplat demonstrate significant speedups by optimizing key computational bottlenecks, achieving substantial reductions in the time required for specific operations compared to baseline approaches [3,6]. Future directions involve exploring more efficient data structures, advanced parallelization strategies, and hardware-specific optimizations [20].

Rendering efficiency is paramount for achieving real-time performance, a key advantage of Gaussian-based methods [2]. The fundamental approach employed is patch-level or tile-based rasterization [1,5,15], which divides the image into tiles (e.g., 16x16 pixels) [5,20] and processes Gaussians intersecting each tile in parallel using GPU capabilities [5,15,20]. Within tiles, visible Gaussians are selected, sorted by depth, and splatted until opacity saturation [20]. View frustum culling is also applied to exclude Gaussians outside the camera's view [7,14,20]. Reducing the overall Gaussian count contributes to rendering performance [2], though this involves a trade-off with visual fidelity. Hardware acceleration, particularly through GPU parallel computing using CUDA, is crucial for real-time performance [5,14,15]. Dedicated hardware solutions like GSCore have also been proposed to specifically optimize the 3DGS rendering pipeline [2]. Achieving optimal rendering efficiency involves balancing these techniques with the preservation of visual quality [29,33].

4.1 Storage Efficiency

A significant challenge in 3D Gaussian Splatting (3DGS) is the high storage overhead, primarily due to the requirement of millions of distinct Gaussian primitives to accurately represent scene geometry and appearance [2]. Reconstructions of typical outdoor scenes can occupy several hundred megabytes to several gigabytes of explicit storage space [2]. This storage burden stems from storing the geometric and appearance attributes of each primitive individually, which can lead to considerable redundancy given the potential similarities among primitives [2]. Consequently, research efforts have focused on developing techniques to enhance storage efficiency.

One prevalent strategy involves applying vector quantization (VQ) to compress the attributes of numerous Gaussian primitives [2]. VQ-based methods, such as EAGLES, are employed to compress Gaussian attributes and subsequently reduce the storage footprint of 3DGS models [29,33]. Compact3D exemplifies this approach by applying VQ to compress different Gaussian attributes into distinct codebooks [2]. Instead of storing the full attributes for each Gaussian, Compact3D stores the index corresponding to the entry in the relevant codebook, thereby reducing storage requirements [2].

Another effective method for storage reduction is employing pruning strategies, which aim to decrease the total number of Gaussian primitives. These strategies typically identify and remove Gaussians deemed less important based on various criteria [2]. LightGaus, for instance, proposes a Gaussian pruning strategy guided by global importance scores [2]. Similarly, CityGaussianV2 reduces storage overhead by pruning Gaussian primitives with low importance scores during the training process [12]. CityGaussianV2 also incorporates a quantization compression strategy specifically for 2DGS, enabling the storage of large-scale scene reconstruction results with a reported overhead of approximately 400 megabytes [12].

When evaluating compression techniques like VQ and pruning, a fundamental trade-off exists between the compression ratio achieved and potential information loss. Higher compression ratios, while reducing model size, may lead to greater information loss, which could negatively impact rendering quality or fidelity. Conversely, minimizing information loss typically results in lower compression ratios. The effectiveness of VQ-based methods like Compact3D and pruning strategies like LightGaus in compressing attributes and reducing the number of Gaussians must be weighed against these potential impacts on visual quality and rendering performance.

Some works explore alternative paradigms beyond traditional non-topological VQ codebooks or simple pruning to address memory concerns [2]. Self-Organizing Gaussians, for example, employs the concept of self-organizing maps to map Gaussian attributes onto corresponding 2D grids [2]. This approach differs from standard VQ by introducing a topological structure to the quantized space. Furthermore, other research focuses on fundamentally improving the effective representation of Gaussians. Techniques such as Scaffold-GS design anchors and additional attributes for more efficient representation, with the capability to convert back to the standard 3DGS format [2]. Similarly, GES introduces generalized exponential functions (GEF) mixing as a replacement for the Gaussian representation, claiming effectiveness in fitting arbitrary signals [2]. These diverse approaches highlight the ongoing efforts to optimize the storage footprint of 3DGS models through various compression, pruning, and representation techniques.

4.2 Training Efficiency

Improving the training efficiency of 3D Gaussian Splatting (3DGS) models is a crucial area of research [2]. Various techniques have been proposed to accelerate the training process and reduce resource utilization. One notable technique, DISTWAR, aims to accelerate atomic operations within rasterization-based differentiable rendering applications, which are fundamental to the 3DGS training pipeline [2]. Beyond specialized operations like those addressed by DISTWAR, the use of custom CUDA kernels and optimized fast rasterization techniques are commonly employed strategies to significantly speed up training [20].

Optimizations to the overall training pipeline also contribute to efficiency. For instance, CityGaussianV2 addresses the challenge of large-scale scenes by optimizing the parallel training pipeline, unifying the training and compression processes. This unification, coupled with the elimination of redundant post-processing steps, streamlines the workflow and likely enhances efficiency, although specific quantitative speedups are not detailed in the provided digest [12].

Some methods demonstrate substantial computational efficiency improvements in specific components. FlashSplat, for example, highlights its ability to compute the necessary sets for its operation in approximately 26 seconds. This is presented as a significant speedup compared to baseline methods which require extensive training time, indicating that optimizing key computational bottlenecks can drastically reduce overall processing time [6].

The challenges in accelerating 3DGS training primarily revolve around the computational intensity of the rasterization and differentiation steps, as well as managing memory bandwidth and parallel processing effectively, especially for large scenes [12]. Potential avenues for further improvement include continued research into more efficient data structures, advanced parallelization strategies, hardware-specific optimizations through custom kernels [20], and novel approaches that reduce the inherent computational complexity of the rendering and optimization process itself.

4.3 Rendering Efficiency

Real-time rendering constitutes a significant advantage of Gaussian-based methods, achieved through various optimization techniques applied to the 3D Gaussian Splatting (3DGS) rendering pipeline [2]. A fundamental approach to enhancing rendering speed and avoiding costly per-pixel computation is patch-level or tile-based rasterization [1,5]. This technique involves dividing the rendered image into non-overlapping tiles, typically measuring 16×16 pixels [5,20]. For each tile, the rendering process identifies the projected 3D Gaussians that intersect with its area [5]. Relevant Gaussians are copied and assigned a tile identifier, facilitating parallel processing [27].

The independence of tile and pixel calculations is leveraged for significant speed improvements [5]. Utilizing CUDA-based blocks and threads, the system enables parallel computation across tiles, enhancing access to shared memory and maintaining an ordered reading sequence [5]. Within each tile’s frustum, visible 3D Gaussians are selected, potentially filtering based on criteria such as a high confidence interval (e.g., greater than 99%) [20]. These selected Gaussians are then sorted by depth and splatted in parallel onto the tile until opacity saturation is reached [20].

Further rendering efficiency is achieved through view frustum culling, which eliminates Gaussians located outside the camera’s field of view before processing [7,20]. Reducing the overall number of Gaussian primitives can also contribute to improved storage efficiency and rendering performance [2]. While methods exist for estimating the impact of primitives based on spatial proximity [2], a detailed analysis of the trade-off between primitive count reduction and visual fidelity is crucial for optimizing this approach.

The benefits of hardware acceleration, particularly through technologies like CUDA which enable the described parallel processing, are central to achieving real-time 3DGS rendering performance [5,15]. Beyond general-purpose parallel computing platforms, dedicated hardware solutions have been proposed, such as GSCore, which aims to optimize the 3DGS rendering pipeline specifically within the context of radiation field rendering [2]. These hardware-specific optimizations contrast with purely software-based approaches by exploiting architectural efficiencies, although they may require specialized hardware infrastructure. The primary rendering optimization approach in 3DGS, as detailed in the digests, relies heavily on parallel, tile-based processing combined with basic culling and sorting, demonstrating a strong dependency on GPU capabilities for its real-time performance [1,5,7].

5. Extensions and Variations

Extension Category	Focus	Key Challenges Addressed	Representative Techniques / Ideas	Examples (Cited)
Handling Dynamic Scenes	Modeling/Rendering non-rigid motion over time	Temporal information, Deformation	4D Spatiotemporal Gaussians, Deformable Gaussians, MLP-based time-varying params, Flow fields	Deformable3DGS, Gaussian-Flow
Large-Scale Scenes	Managing memory & computation for vast environments	Memory explosion, Training instability	Scene Partitioning (Sub-models), Hierarchical Struct., Data Allocation	CityGaussianV2

Improving Quality	Enhancing realism, reducing artifacts, refining geometry	Fine details, Sharp edges, Artifacts, Geometry	Sophisticated Gaussian Reps (SH, Materials), External Data/Supervision (Depth, Normals), Geometry Refinement (Densification, Filtering)	GaussianShader, R3DG, Multi-scale gs, MIP-Splting, CityGaussianV2, GaussianPro, FreGS, DeblurGS

Extensions and variations of the foundational 3D Gaussian Splatting (3DGS) technique address key limitations and expand its applicability to more complex scenarios beyond static, small-to-medium scale scene reconstruction. These advancements primarily focus on handling dynamic content, managing large-scale environments, and enhancing the overall quality of the synthesized views and reconstructed geometry [2,12,24].

Handling dynamic scenes necessitates incorporating temporal information and modeling non-rigid deformations, as the base 3DGS is designed for static environments [14,24]. This often involves extending the Gaussian representation into a 4D spatiotemporal domain [5,29]. Approaches include utilizing deformable 3D Gaussians that change position, scale, and rotation over time to track scene motion [32,33]. This allows for high-fidelity monocular dynamic scene reconstruction by modeling deformation fields [14]. Alternative methods leverage position encoding combined with Multi-Layer Perceptron (MLP) networks to predict time-varying Gaussian properties or employ other explicit modeling techniques [29,33]. Specific implementations like Deformable3DGS demonstrate deformable models for non-rigid motion, while Gaussian-Flow addresses limitations in earlier dynamic 3DGS techniques to improve reconstruction quality and stability [33,34]. Key methods for dynamic scenes involve flow field-based deformation, neural network-based dynamic modeling, and spatiotemporal consistency constraints [14].

Extending 3DGS to large-scale scenes introduces significant challenges, primarily related to memory consumption and computational complexity [29]. Dense representation of vast environments with a large number of Gaussians can quickly exceed GPU memory limits. Solutions often involve scene partitioning, dividing the environment into smaller, manageable sub-regions or sub-models that can be processed and stored independently [12]. Efficient data allocation schemes load only relevant sub-models based on the camera's viewpoint. CityGaussianV2 exemplifies this approach, employing sub-model partitioning and data allocation for large urban environments and utilizing 2DGS as a base element for reconstruction to enhance efficiency [12]. Hierarchical structures are also explored as potential solutions to the memory explosion issue inherent in large-scale applications [12,33].

Improving the quality of reconstructed scenes is achieved through various means, including using sophisticated Gaussian representations, integrating additional data or supervision, and refining geometry [2]. Sophisticated representations like spherical harmonics (SH) capture view-dependent colors for non-Lambertian effects [5]. More advanced methods, such as GaussianShader and Re-illuminable 3D Gaussians (R3DG), explicitly model complex material properties like reflection and re-illumination, respectively [2]. For dynamic scene reconstruction, Gaussian attributes can be updated based on mesh deformation using formulas related to vertex displacement Δv, deformation gradient T, and rotation R derived from polar decomposition [8]:

$$\begin{aligned} v' &= v + \Delta v, \\ \Sigma' &= R\,\Sigma\,R^T, \\ SH(d') &= SH(d + T) \end{aligned}$$

where v and Σ are the spatial position and covariance matrix before deformation, and SH(*) denotes the spherical harmonic function [8]. Techniques like Multi-scale gs and MIP-Splting address rendering quality at different scales and mitigate aliasing through multi-scale Gaussians or frequency constraints [2]. Incorporating additional data or supervision, such as depth regression supervision (e.g., in CityGaussianV2 [12]), depth-guided novel view mask rendering [6], or utilizing a joint 2D-3D training paradigm like GaussianPro to reduce reliance on SfM initialization and enforce consistency of depth and normal maps, helps improve accuracy [2,33]. DeblurGS addresses challenges from inaccurate camera poses by estimating motion and optimizing the scene accordingly [2]. Geometry refinement is crucial, involving densification schemes (e.g., in

CityGaussianV2 [12]), progressive Gaussian propagation based on 3D plane definition and patch matching (e.g., in GaussianPro [2]), geometric filtering, sparse control, and frequency-domain methods like FreGS that use Fourier transforms to mitigate over-reconstruction and eliminate artifacts [2,32].

In comparing these extensions, dynamic scene techniques are effective for capturing temporal changes but require robust motion modeling. Large-scale methods primarily address memory limitations through spatial decomposition, essential for urban or expansive environments. Quality improvement methods tackle fidelity issues by enriching the Gaussian representation, leveraging external geometric cues, or refining the optimization process. Each category of extension targets specific challenges, demonstrating effectiveness within its domain. Dynamic methods are crucial for applications involving movement, large-scale methods for expansive environments, and quality improvements are universally beneficial for enhancing realism and geometric accuracy across applications [2,12,24,33]. While significant progress has been made, challenges remain in achieving robust, high-quality, and efficient reconstruction for highly dynamic and extremely large-scale scenes simultaneously, suggesting avenues for future research focusing on integrated solutions that combine sophisticated modeling, efficient data management, and advanced optimization techniques.

5.1 Handling Dynamic Scenes

Extending 3D Gaussian Splatting (3DGS) to effectively model and render dynamic scenes presents a significant challenge, as the standard formulation is primarily designed for static environments [18]. Addressing this involves techniques capable of representing and evolving scene geometry, appearance, and motion over time [24]. One conceptual extension introduces a temporal dimension to the Gaussian representation—often referred to as 4D Gaussian Splatting—thereby enabling the expression and rendering of dynamic content [5,29].

Several approaches have been developed to capture and represent these temporal changes. A prominent technique involves utilizing deformable 3D Gaussians, which allow the individual Gaussians to change their position, scale, and rotation over time to conform to the dynamic scene structure [32,33]. This method facilitates high-fidelity monocular dynamic scene reconstruction by modeling the deformation fields. Beyond explicit deformation modeling, other strategies include methods based on position encoding combined with Multi-Layer Perceptron (MLP) networks to predict Gaussian properties over time, as well as alternative explicit modeling techniques that track or represent changes differently [29,33]. Specific implementations, such as Deformable3DGS, exemplify the application of deformable models for handling non-rigid motion [33]. Furthermore, methods like Gaussian-Flow have been introduced to specifically address some of the limitations inherent in earlier dynamic 3DGS techniques, aiming to improve reconstruction quality and stability for dynamic content [34]. The development of these diverse modeling techniques—including deformable models, MLP-based approaches, and explicit tracking or representation methods—collectively contributes to extending the applicability of 3DGS from static scene representation to the complex domain of dynamic scene capture and rendering.

5.2 Large-Scale Scene Reconstruction

Applying 3D Gaussian Splatting (3DGS) to large-scale environments represents a significant advancement in the field of novel view synthesis and scene reconstruction [29]. However, reconstructing expansive scenes introduces substantial challenges, primarily related to memory consumption and computational complexity. The dense representation of large environments using millions or billions of 3D Gaussians can quickly exhaust available GPU memory and lead to prohibitively long processing times for optimization and rendering.

To address these challenges, researchers have explored techniques such as scene partitioning and hierarchical methods. Scene partitioning involves dividing the large scene into smaller, manageable sub-regions or sub-models. This allows each sub-region to be processed and stored independently, significantly reducing the peak memory required compared to handling the entire scene as a single unit. Data allocation schemes are designed to efficiently load and unload these sub-models based on the camera's viewpoint, ensuring only relevant portions of the scene are active during rendering.

An example of this approach is demonstrated by CityGaussianV2, which is specifically designed for large-scale scene reconstruction [12]. CityGaussianV2 inherits a sub-model partitioning and data allocation scheme, enabling the reconstruction of vast urban environments by breaking them down into smaller, more manageable components [12]. Furthermore, CityGaussianV2 employs 2DGS as the base element for reconstruction in large-scale scenes, suggesting an adaptation of the fundamental representation or rendering process to enhance efficiency for this challenging scale [12]. This strategy of partitioning and specialized data handling is crucial for overcoming the memory explosion problem inherent in applying standard 3DGS to large-scale data. While hierarchical methods are also posited as potential solutions, the partitioning approach exemplified by systems like CityGaussianV2 highlights a key strategy for making 3DGS feasible for reconstructing and rendering large environments.

5.3 Improving Reconstruction Quality

Improving the quality of reconstructed scenes is a significant focus within 3D Gaussian Splatting research, aiming to reduce artifacts and enhance the realism and accuracy of novel view synthesis [2]. This is achieved through various approaches, including the use of more sophisticated Gaussian representations, the incorporation of additional data or supervision, and refinements in geometry.

Sophisticated Gaussian representations play a crucial role in capturing scene complexity. For instance, representing view-dependent colors using spherical harmonics (SH), initially seen in Plenoxels, enables the depiction of non-Lambertian effects such as specular reflections, in contrast to simpler RGB representations [5]. Further advancements address complex material properties explicitly. GaussianShader, for example, reconstructs reflective surfaces using a mixed color representation that includes diffuse, direct specular reflection, and residual components, alongside integrated specular GGX and normal estimation modules [2]. Similarly, Re-illuminable 3D Gaussians (R3DG) represent scenes with re-illuminable points characterized by normal direction, BRDF parameters, and decomposed incident illumination components [2]. For dynamic scenes, the attributes of Gaussian spheres, including covariance matrix, spatial position, and spherical harmonic input perspective, can be updated based on the deformation gradient and displacement of corresponding mesh vertices. This deformation process is governed by the following formulas [8]:

$$\begin{aligned} v' &= v + \Delta v \\ \Sigma' &= R \Sigma R^T \\ SH(d') &= SH(d + T) \end{aligned}$$

where v and Σ are the spatial position and covariance matrix before deformation, $SH(*)$ is the spherical harmonic function, Δv is vertex displacement, T is the deformation gradient, and R is derived from the polar decomposition of T. These updates allow Gaussians to accurately represent the deformed scene geometry and appearance [8]. Handling rendering at different scales is also addressed; Muti-scale gs analyzes frequency-domain aliasing at low resolutions and distances and uses multi-scale Gaussians to mitigate it, while MIP-Splting applies a Gaussian low-pass filter based on the Nyquist theorem to constrain 3D Gaussian frequencies [2].

Incorporating additional data or supervision signals helps constrain the reconstruction process and improve accuracy. CityGaussianV2 introduces depth regression supervision to provide geometric priors, aiding reconstruction quality [12]. The use of depth-guided novel view mask rendering can generate 2D masks for previously unseen views [6]. GaussianPro leverages a joint 2D-3D training paradigm to reduce dependence on SfM initialization [2]. This approach includes a progressive Gaussian propagation strategy utilizing the consistency of 3D views and projection relationships to optimize rendered 2D depth and normal maps [2]. Optimized depth and normal maps are subsequently used for densification and supervision to achieve precise geometric representation [2]. Furthermore, GaussianPro leverages normal direction consistency between adjacent views for better reconstruction results [33]. Techniques like introducing a background bias also help reduce noise in segmentation results, contributing to improved quality [6]. Addressing challenges from inaccurate camera poses due to blurring, DeblurGS estimates 6-DoF camera motion and synthesizes corresponding blurred renderings to optimize the scene [2].

Refining geometry is another critical aspect. CityGaussianV2 proposes a new Gaussian primitive densification scheme to enhance reconstruction quality [12]. GaussianPro's progressive Gaussian propagation, based on 3D plane definition and patch matching, facilitates geometric refinement [2]. This includes a geometric filtering and selection process [2]. Techniques operating in the frequency domain, such as FreGS, transform supervision to the frequency domain, utilizing amplitude and phase properties of 2D discrete Fourier transforms to mitigate over-reconstruction and introduce a frequency-domain-guided coarse-to-fine annealing technique to eliminate artifacts [2]. Sparse control of Gaussians and progressive propagation are also mentioned as techniques to enhance rendering quality [32]. These geometrical refinements, often guided by explicit geometric cues or constraints, contribute significantly to reducing artifacts and improving the structural accuracy of the reconstruction. Collectively, these methods, ranging from enriched Gaussian properties to leveraging external data and geometric constraints, contribute to improving the photorealism and accuracy of 3D Gaussian Splatting reconstructions [2,33].

6. Applications

The advent of 3D Gaussian Splatting (3DGS) has catalyzed significant advancements across a multitude of application domains within computer graphics and computer vision [14,18,23,25,28,29,33]. Its explicit representation, coupled with highly efficient differentiable rendering, enables high-quality scene reconstruction and novel view synthesis with real-time performance, distinguishing it from prior neural rendering techniques and traditional methods [5,16,31]. This combination of fidelity and speed makes 3DGS a versatile tool with broad impact.

One primary application area is **Novel View Synthesis and 3D Reconstruction**. 3DGS excels at generating photorealistic images from arbitrary viewpoints of a captured scene [4,16]. Its anisotropic Gaussians effectively represent scene structure and fine details, including complex lighting, often achieving state-of-the-art visual quality and significantly faster training and rendering times compared to previous neural methods like NeRF variants [5,11]. Furthermore, 3DGS is adept at 3D reconstruction from various inputs, including sparse or even single views, and can handle large-scale environments efficiently [4,29,33]. It also facilitates downstream tasks like generating object masks and efficient mesh extraction, demonstrating its utility beyond just rendering.

Another significant domain is **Scene Editing and Manipulation**. The explicit nature of the Gaussian representation allows for intuitive and efficient modification of 3D scenes, offering advantages over implicit and traditional mesh-based methods [25]. This includes geometry editing, such as 3D completion, object removal, interactive manipulation, and large-scale deformation, often guided by text prompts or semantic information [6,8,29,33]. Appearance editing, including text-based modifications and texture/lighting separation, is also supported. Tools like GaussianEditor streamline these processes, enabling rapid scene modifications [10,28].

SLAM, Robotics, and Autonomous Driving benefit substantially from 3DGS due to its real-time performance, accuracy, and ability to represent environmental details [5,24]. Accurate, real-time 3D environment perception and mapping are critical for navigation and interaction in these fields [21]. 3DGS-based systems like GS-SLAM, SplaTAM, and Gaussian Splatting SLAM demonstrate efficient and accurate tracking and dense mapping, even with monocular cameras, offering superior performance trade-offs compared to traditional point cloud or voxel methods [17,30]. In autonomous driving, 3DGS is applied for scene reconstruction, including handling dynamic elements and integrating diverse data sources like LiDAR [5,24].

In **VR, AR, and Interactive Media**, the ability of 3DGS to deliver high-quality, real-time rendering of complex scenes is paramount for creating immersive and comfortable user experiences [14,25]. It supports applications like real-time AR scene reconstruction on mobile devices and the creation of interactive virtual environments [18]. While challenges remain in handling dynamic lighting in standard 3DGS models, its core performance provides a strong foundation for future advancements in these highly interactive domains.

Beyond these core areas, 3DGS is enabling various **Other Applications**. This includes sophisticated **3D Content Generation** (text-to-3D/4D, image-to-3D) often integrating diffusion models and physics simulations for dynamic or physically realistic content [10,28,29,33]. **Semantic Segmentation** in 3D scenes is significantly accelerated by combining 3DGS with 2D scene understanding models, enabling interactive and multi-granularity segmentation [28,29,33]. It is also widely used in **Digital Human** modeling for reconstruction and animation of full bodies, heads, hair, and hands [2,29,33]. In **Content Creation and Visual Effects**, 3DGS serves as a powerful tool for digital twins, architectural visualization, film/game scene construction, simulating natural phenomena, and processing geographic data [4,18,22]. Emerging applications even include data hiding within 3DGS models and its potential for holographic communication [13,34].

Collectively, the diverse applications of 3DGS underscore its disruptive potential in fields reliant on 3D data acquisition, representation, and rendering [25]. Its efficiency, quality, and editability are driving innovation and lowering barriers to 3D content creation and interaction, suggesting a transformative impact on future technologies [5,22].

6.1 Novel View Synthesis and 3D Reconstruction

Three-dimensional Gaussian Splatting (3DGS) has emerged as a significant technique in the fields of novel view synthesis and 3D reconstruction [25]. The method represents a 3D scene using a set of anisotropic 3D Gaussians, each characterized by position, covariance, opacity, and spherical harmonic coefficients for color [10,28]. The anisotropic covariance allows for an accurate representation of the scene structure, while a fast visibility-aware rendering algorithm projects these 3D Gaussians onto the 2D image plane to synthesize novel views [10,28]. This combination facilitates high-quality, real-time rendering [10,16,19,20,32,34].

In novel view synthesis, 3DGS has demonstrated state-of-the-art performance in terms of visual quality [8,10,16,19,20,26,32,34]. It surpasses prior neural rendering techniques such as Mip-NeRF360, InstantNGP, and Plenoxels in rendering quality [26]. Furthermore, 3DGS excels at capturing fine details and complex lighting effects, leading to highly photorealistic results [4]. Methods like HDR-GS apply 3DGS to high dynamic range (HDR) scene rendering, achieving state-of-the-art PSNR performance and significantly reducing training and inference times compared to methods like HDR-NeRF, enabling the rendering of richer image details and better capture of HDR scenes [11]. Beyond quality, 3DGS offers notable improvements in speed, providing fast training and inference times [11,19,32]. This combination of high fidelity and real-time performance makes 3DGS suitable for applications requiring interactive rendering [16,20]. The representation can also support downstream tasks, such as generating 2D masks for objects in novel views [6].

For 3D reconstruction, 3DGS demonstrates versatility across various input modalities [25,29,32,33,34]. It is effective in reconstructing 3D scenes from sparse view inputs, where traditional photogrammetry methods may struggle [29,33]. For example, GaussianObject achieves high-quality reconstruction from as few as four input views [33]. Reconstructing scenes from single-view inputs is also possible; Splatter Image is an ultra-fast single-view method based on Gaussian splatting that achieves 3D reconstruction at 38 FPS by mapping each pixel of the input image to a 3D Gaussian using a 2D image-to-image network [10,28]. An extension of this approach addresses multiple input images using cross-view attention [10]. Furthermore, 3DGS is capable of handling large-scale scenes, efficiently representing complex environments with millions of splats without significant performance degradation [4,33]. CityGaussianV2, for instance, demonstrates excellent geometric accuracy and comparable rendering quality for large-scale streetscapes, enabling realistic browsing experiences [12]. Beyond generating dense point representations, methods like SuGaR facilitate fast mesh extraction from 3D Gaussian splatting. SuGaR employs a regularization term to encourage Gaussians to align with the scene surface and utilizes Poisson reconstruction to extract the mesh, with an optional refinement step that binds Gaussians to the mesh surface and co-optimizes them [10,28]. Specific applications like GPS-Gaussian highlight the capability for real-time human novel view synthesis, suitable for applications such as holographic communication [34].

In summary, 3DGS provides a powerful framework for both high-quality novel view synthesis with real-time performance and versatile 3D reconstruction from various inputs including sparse and single views [16,25,29,33]. Its ability to capture fine details, handle complex lighting, represent large-scale environments, and support efficient mesh extraction demonstrates its broad applicability and state-of-the-art capabilities in these fundamental 3D vision tasks [4,10,12,28].

6.2 Scene Editing and Manipulation

Three-dimensional Gaussian Splatting (3DGS) has significantly advanced the capabilities for editing and manipulating novel view synthesis scenes, offering a level of control and efficiency often challenging with implicit representations or traditional mesh-based methods [25]. The explicit nature of 3D Gaussians, which are discrete primitives characterized by properties such as position, scale, rotation, color, opacity, and spherical harmonics, facilitates direct manipulation and semantic understanding, thereby enabling intuitive and efficient editing operations.

Various techniques have been developed leveraging the 3DGS framework for scene editing. Geometry editing, for instance, can be achieved through diverse approaches, including utilizing text prompts and semantic information for tasks such as 3D completion and object removal [29,33]. Methods also support interactive scene object manipulation and the completion of exposed areas [33]. Specific tasks like object removal and object inpainting have demonstrated effectiveness within this paradigm [6]. For large-scale geometric deformations, pipelines have been proposed that bind 3D Gaussians to meshes, allowing for complex manipulations [33]. Alternatively, users can edit control vertices to achieve high-fidelity deformation modeling of the Gaussian representation [8]. The explicit representation also facilitates the editing of dynamic objects, a capability crucial for applications in content creation and scene understanding [32].

Beyond geometry, 3DGS also supports appearance editing. Techniques enable modifying images based on text prompts, separating textures and lighting for independent editing, and employing neural networks for inverse rendering to understand scene properties [29]. Furthermore, the discrete particle-like nature of Gaussians has been explored for physics simulation, allowing for physics-based dynamics and photorealistic rendering using discrete particle clouds [29].

Several dedicated tools and algorithms have emerged to streamline 3DGS-based editing. GaussianEditor is presented as an efficient 3D editing algorithm that enhances precision and controllability through Gaussian semantic tracking and employs hierarchical Gaussian Splatting for stable and refined results [10,28]. This tool simplifies 3D scene editing, allowing users to perform operations such as object deletion and integration [10] and generally modify 3D scenes rapidly, reportedly in minutes [17,30].

The efficacy and intuitiveness of these editing techniques are significantly attributed to the explicit representation of 3DGS. Unlike implicit neural representations which require complex optimization to alter scene properties, or traditional meshes that can struggle with detailed appearance and transparency manipulation, 3D Gaussians allow for direct selection, transformation, or modification of attributes of individual or groups of primitives. This direct access, combined with efficient differentiable rendering, makes tasks like object segmentation, deletion, and property adjustments more straightforward and computationally less intensive for certain operations, leading to faster workflows and enhanced controllability. The ability to bind Gaussians to semantic labels or control structures further increases the precision and scope of possible manipulations.

6.3 SLAM, Robotics, and Autonomous Driving

Accurate and real-time 3D environment understanding is fundamental for Simultaneous Localization and Mapping (SLAM) systems, robotics, and autonomous driving platforms [27]. These applications require devices to simultaneously determine their position and build a map of their surroundings [27]. Traditionally, SLAM and 3D perception systems relied on representations such as point clouds, surfel clouds, or voxel grids [27]. The advent of 3D Gaussian Splatting (3DGS) has introduced a new paradigm, offering significant advantages in terms of efficiency, precision, and adaptability compared to these conventional methods [27].

The explicit geometry representation provided by 3DGS is particularly beneficial in SLAM, helping to reduce misalignments between different viewpoints [29]. The use of 3D Gaussians as the sole 3D representation enables accurate and efficient tracking, mapping, and high-quality rendering [10,28]. Several 3DGS-based SLAM systems have been proposed, including GS-SLAM, SplaTAM, Gaussian Splatting SLAM, Gaussian-SLAM, and Photo-SLAM [21].

GS-SLAM is notable as one of the initial works to integrate 3D Gaussian Splatting into a SLAM framework, balancing efficiency with accuracy [17,30]. It leverages real-time differentiable splatting rendering to accelerate map optimization and RGB-D re-rendering, achieving rendering speeds up to 100 times faster than previous state-of-the-art algorithms [17,30]. SplaTAM demonstrates the capability of achieving dense reconstruction using a monocular RGB-D camera, reportedly outperforming prior state-of-the-art methods [17,30]. Gaussian Splatting SLAM, particularly the method from Imperial College London, showcases the potential for dense mapping using only a monocular camera, without reliance on Structure-from-Motion (SFM) or learning-based priors [17,30]. This system is capable of generating accurate reconstructions with excellent rendering quality, even for challenging elements like tiny or transparent objects [17,30]. It performs online dense map generation at approximately 3 frames per second (fps) [17], directly optimizing camera tracking against the 3D Gaussians and employing geometric verification and regularization to address potential ambiguities [10].

Beyond SLAM, 3DGS is also being applied in autonomous driving and robotics for real-time 3D environment perception [18]. It facilitates scene understanding and mapping, which is crucial for navigation and interaction [21,32]. Systems like Street Gaussians and DrivingGaussian are examples of its application in this domain [21]. Furthermore, 3DGS enables the construction of high-fidelity virtual testing environments, which are valuable for simulation in the development and validation of autonomous systems [18].

In comparison to traditional point-based or voxel-based methods, 3DGS-based systems offer superior trade-offs in terms of real-time performance and reconstruction quality, driven by efficient rendering and the ability to represent scene details at varying scales [17,27,30]. This makes 3DGS a promising representation for future advancements in SLAM, robotics, and autonomous navigation.

6.4 VR, AR, and Interactive Media

3D Gaussian Splatting (3DGS) significantly contributes to the advancement of virtual reality (VR), augmented reality (AR), and interactive media applications [4,25]. A primary requirement for immersive VR and AR experiences is real-time interaction and rendering, as any notable delay can induce discomfort or simulation sickness in users [1]. 3DGS addresses this need by providing enhanced realism and performance in these environments [4]. Its capability to render complex scenes captured from the real world at high frame rates makes it particularly suitable for applications where users need to perceive and interact with virtual or augmented content without perceptible latency.

Specific applications demonstrate the utility of 3DGS in this domain. For example, 3DGS facilitates real-time AR scene reconstruction on mobile devices, enabling high-quality representations of the user's environment and supporting seamless interaction between virtual objects and the real world [18]. Beyond simple reconstruction, 3DGS can be integrated into the creation of interactive virtual environments. A notable instance is the development of a virtual interactive jazz drum kit by Bellevue College's XR LAB [22]. This project utilized 3DGS by capturing video of a physical drum kit from multiple viewpoints, processing and training the data using Postshot, and exporting the resulting .ply file. Subsequent post-production in platforms like PlayCanvas and Spline allowed for the refinement of the point cloud, as well as the implementation of interactions, interface design, and sound effects [22]. This example highlights the pipeline integration potential of 3DGS for creating specific, interactive assets or scenes within larger media projects.

Despite its strengths in real-time rendering and realism, the conventional 3DGS approach presents limitations that impact its application in highly dynamic interactive media. Specifically, the inability of standard 3DGS models to change the brightness or lighting conditions of the rendered scene restricts their use in applications such as AR, VR, movies, and games, where dynamic lighting is often essential for conveying mood, environmental changes, or realistic interaction effects. Addressing this limitation represents a key area for future research and development to unlock the full potential of 3DGS in creating truly dynamic and immersive interactive experiences. Future directions in this field are likely to focus on integrating techniques for dynamic lighting, supporting complex material properties, and enabling more sophisticated object manipulation and interaction within the splatted scenes in order to meet the evolving demands of VR, AR, and interactive content creation.

6.5 Other Applications

The explicit nature and real-time rendering capabilities of 3D Gaussian Splatting (3DGS) have enabled its application across a diverse range of fields beyond novel view synthesis. These domains leverage 3DGS for tasks such as object and scene generation, interactive editing, simulation, and digital content creation.

One significant area is text-to-3D generation, exemplified by methods like GSGEN [10,28]. GSGEN utilizes a progressive optimization strategy involving geometry optimization and appearance refinement to generate high-quality 3D objects from textual prompts, incorporating a 3D prior for detailed results [10]. The broader field of 3D/4D generation also benefits from 3DGS, often integrating diffusion models for text-to-3D/4D and image-to-3D synthesis [29,33]. Techniques like Score Distillation Sampling (SDS) are crucial in AI-generated content (AIGC) for creating 3D representations with multi-view consistency [2]. Some research focuses on feedforward network-based generation, bypassing the need for scene-specific training, while others explore generating entire 3DGS scenes from a single image [2]. For dynamic scene generation, methods like Dynamic 3D Gaussians model scenes using sets of Gaussians that move and rotate over time, enabling simultaneous novel view synthesis and 6-DOF tracking under local rigid body constraints [10,28]. Integrating physics-based dynamics, PhysGaussian uses a custom Material Point Method (MPM) to endow 3D Gaussians with physically meaningful motion and mechanical stress properties for high-quality novel motion synthesis [10,28].

Semantic segmentation of 3D scenes is another prominent application. 3DGS allows for real-time 3D scene representation by combining 2D scene understanding methods with the 3D Gaussian structure [33]. Methods like Segment Any 3D Gaussians (SAGA) integrate 2D segmentation foundation models, such as SAM and those leveraging CLIP/DINO for dense language features, with 3DGS to achieve multi-granularity interactive 3D segmentation [10,28,33]. This fusion, sometimes via contrastive training [10], enables adaptation to various prompts and offers significant speed advantages, potentially performing 3D segmentation in milliseconds with substantial acceleration compared to traditional methods [17].

In the realm of digital humans, 3DGS supports tasks like human reconstruction, animation, and generation [2,29,33]. Specific techniques address detailed aspects, such as GaussianAvatars which integrates FLAME grids as prior knowledge for enhanced head reconstruction quality [2]. Gaussian Hair employs linked cylindrical Gaussians specifically for detailed hair reconstruction [2]. Beyond heads and hair, 3DGS facilitates efficient full-body and hand modeling [33]. The editable nature of 3DGS makes it highly relevant for creating and manipulating avatars [5,27].

Content creation and visual effects widely benefit from 3DGS. Its capabilities are applied in digital twins and virtual reality, including rapid creation of high-fidelity building models for architectural visualization, real-time visualization of factory layouts, and digitization of cultural relics [18]. The film and game industries utilize 3DGS for rapid scene construction, offering an alternative to traditional 3D modeling pipelines, and for simulating complex natural phenomena like smoke and clouds in special effects [18]. Geographic Information Systems (GIS) also leverage 3DGS for fast processing of dronecollected data for urban modeling and large-scale terrain visualization [18]. The accessibility of user-friendly 3D modeling software and mobile applications based on 3DGS, often utilizing cloud computing, lowers the barrier to entry for users to create 3D models from simple video input [22].

Emerging applications showcase the versatility of 3DGS. In security and steganography, GS-Hider allows embedding 3D scenes and images into original GS point clouds invisibly [13]. This framework replaces spherical harmonics coefficients with a coupled secured feature attribute, using separate decoders to retrieve the original scene and the hidden message [13]. Other applications include autonomous driving, where 3DGS can reconstruct coherent scenes by mixing data points like LiDAR, aiding in handling varying densities and accurately modeling static and dynamic elements [5,27]. The potential for 3DGS in holographic communication has also been noted [34]. While specific details are not provided in the digests, the subsection description also references the application of 3DGS in HDR imaging [11].

Overall, 3DGS's unique properties make it a powerful tool enabling new capabilities across various domains, from generative AI and digital content production to specialized applications like security and environmental perception for autonomous systems. Its rapid evolution suggests a growing impact on how 3D data is created, processed, and utilized.

7. Challenges and Future Directions

Despite the significant advancements demonstrated by 3D Gaussian Splatting (3DGS) in novel view synthesis and real-time rendering, the technology still faces several inherent limitations and technical challenges that require ongoing research [14,25,29,33]. Addressing these challenges is crucial for expanding its applicability and enhancing its performance in more complex and demanding scenarios. The primary limitations and challenges can be categorized as follows:

Challenge Category	Description	Specific Issues	Impact
Gaussian Representation Limits	Inherent limitations of using anisotropic Gaussians	Precise Geometric Fidelity, Representing Fine Details/Edges, Handling Reflective/Untextured Surfaces	Artifacts, Lack of accuracy vs. meshes, Features lost/jagged
Complex & Dynamic Scenes	Difficulty in handling intricate static geometry and motion over time	Intricate Geometry/Occlusions, Scaling to Large Scenes (Convergence, Stability, Training), Dynamic Objects/Non-rigid Motion Support	Reduced Quality, Efficiency issues, Limited applicability
Efficiency, Memory, Scalability	High resource requirements	Storage Size (GB vs MB for NeRF), Memory for Large Scenes, Training Time/Memory, Optimization on resource-constrained platforms	Scalability limits, Hardware dependency, Vulnerability to attacks
Data Efficiency & Generalization	Reliance on dense multi-view input	Data Sparsity, Sparse/Limited Views, Artifacts in undersampled areas	Incomplete Recon, Reduced Fidelity, Limited Applicability

Rendering Quality & Artifacts	Visual inaccuracies in rendered output	Inaccurate Depth Ordering, Sub-optimal Alpha Blending, Aliasing	Reduced Realism, Visual Distractions
Security Concerns	Vulnerability to malicious attacks	Data Poisoning (e.g., Poison-Splat) increasing computation/memory	System Overload, Service Disruption

First, challenges related to the **Limitations of the Gaussian Representation** itself exist. While anisotropic Gaussians are effective for many scenes, they can struggle with precise geometric fidelity compared to high-quality mesh-based representations [18]. Representing extremely fine details, sharp edges, and thin structures accurately remains difficult; such features may appear jagged or be lost [4]. Additionally, reconstructing scenes with features like reflective surfaces or untextured walls (common in indoor environments) presents challenges for the standard Gaussian representation [15].

Second, **Handling Complex and Dynamic Scenes** poses significant hurdles. Complex static scenes with intricate geometry and occlusions are challenging [37]. Scaling 3DGS to large environments introduces issues such as slow convergence, training instability, and difficulties in parallel training efficiency, impacting reconstruction quality [12]. Furthermore, extending 3DGS to effectively model and render dynamic scenes with complex, non-rigid motion over time is a major challenge, as current methods have limited support for dynamic objects [14,18,29,33].

Third, **Efficiency, Memory, and Scalability** remain critical concerns. 3DGS models often require significant storage space, typically in gigabytes, which is considerably more than many NeRF models that can be stored in megabytes [4,7]. This substantial memory footprint affects scalability, making efficient reconstruction and rendering of very large scenes challenging [2,14,24]. Training time and memory consumption during optimization also contribute to efficiency issues, exacerbated by potential vulnerabilities [9]. While real-time rendering has been achieved, optimization for resource-constrained platforms like mobile devices is still needed [14].

Fourth, challenges exist regarding **Data Efficiency and Generalization**. 3DGS relies on multi-view input, leading to difficulties with data sparsity, especially when the number of views is limited or the viewpoints are sparse [2,18]. This sparsity can cause artifacts and incomplete reconstructions in undersampled areas [24]. Achieving robust, data-efficient training and ensuring models generalize well from limited views or to new scenes are active research areas [2].

Fifth, ensuring high **Rendering Quality and Addressing Artifacts** is an ongoing challenge. Issues such as inaccurate depth ordering and sub-optimal alpha blending of Gaussians can lead to visible artifacts [2]. Aliasing is another common artifact, although methods like Mip-Splatting attempt to mitigate this by constraining Gaussian sizes and using a 2D Mip filter [10]. Developing more sophisticated rendering algorithms is necessary to eliminate these artifacts and enhance realism [2].

Finally, emerging **Security Concerns** highlight potential vulnerabilities. The Poison-Splat attack, for instance, is a computation cost attack targeting the training process by perturbing input images, which can drastically increase memory usage, training time, and Gaussian count, potentially leading to system overload and server downtime [9]. Existing security measures like steganography for protecting assets post-training do not address such input data poisoning vulnerabilities [13].

Research Direction	Focus	Specific Goals	Potential Techniques / Ideas
Improve Data Efficiency	Reduce reliance on dense multi-view input	Few-shot Learning, Handle Sparse Areas, Interpolation/Integration	Depth info, Dense Probability Distributions, Pixel-to-Gaussian Mapping
Enhance Storage & Scalability	Reduce model size, improve	More Efficient Data Structures, Compression,	Light-Gaussian, Hierarchical Structures

	performance for large scenes	Memory Optimization	
Robust Dynamic Scene Handling	Accurately model and render changing scenes	Robust non-rigid motion tracking/rendering	Advanced Spatiotemporal Models
Increase Robustness	Enhance resilience to input data issues and malicious attacks	Noise/Occlusion Resilience, Defenses vs. Data Poisoning (e.g., Poison-Splat)	Intelligent Detection/Countermeasures
Improve Editing & Artistic Control	Make scene modification easier and more intuitive	Intelligent Editing Tools, Greater Control over Appearance/Geometry	Semantic-guided editing, Physics integration
Handle Complex Lighting	Accurately represent and render challenging lighting conditions	Improve HDR scene quality, Dynamic lighting	Inverse Rendering, Material Modeling
Improve Generalization	Enable models to perform well on new scenes/conditions without retraining	Robustness to varying inputs/scenes	Few-shot generalization, Domain Adaptation techniques
Integrate with Other Techniques	Combine 3DGS with other vision/graphics methods	Leverage strengths of other methods, Hybrid approaches	Mesh reconstruction from Gaussians, Cross-modal learning, Joint optimization

These challenges pave the way for numerous **Open Problems and Promising Future Research Avenues** [2,14,29,33]. Key directions include improving data efficiency, such as few-shot learning and methods for handling sparse areas through interpolation or integration [5,24]. Addressing storage efficiency and scalability for large scenes through more efficient data structures, compression techniques like Light-Gaussian, and memory optimization is crucial [5,6,24]. Extending the technology to robustly handle dynamic scenes is a significant area of future work [6,14]. Enhancing robustness to noise, occlusions, and specifically developing intelligent defenses against data poisoning attacks like Poison-Splat are critical for reliability [6,9]. Improving editing and artistic control capabilities is essential for creative workflows [4,14]. Further research is needed to handle complex lighting effects and HDR scenes [11]. Other directions involve improving generalization ability [14], integrating with other rendering or vision techniques [4,14], exploring cross-modal learning [14], and investigating the potential of 3D Gaussians for mesh reconstruction [24]. Ultimately, future work aims to lower technical barriers for creators, enabling richer content generation, and expand applications in areas like AR, VR, and content creation [11,22].

7.1 Limitations of the Gaussian Representation

While 3D Gaussian Splatting (3DGS) offers significant advantages in novel view synthesis, the fundamental representation of scenes using anisotropic Gaussians introduces inherent limitations [37]. These limitations frequently manifest as various artifacts and inaccuracies in both the reconstructed 3D scene structure and the final rendered images [15]. A notable limitation pertains to the geometric fidelity of the representation; specifically, the geometry accuracy achieved by 3DGS is generally not as high as that attainable with detailed, high-quality mesh-based representations [18]. Furthermore, the splat representation can encounter difficulties in accurately preserving extremely fine details within a scene. Such minute features may occasionally be lost during the optimization and rendering process, or they can appear with undesirable jaggedness in the rendered output [4]. These issues contribute to the overall quality of the scene representation and rendering, highlighting areas where the Gaussian representation may fall short compared to alternative methods or ideally accurate depictions.

7.2 Handling Complex and Dynamic Scenes

Handling complex scenes with intricate geometry and significant occlusions presents notable challenges for 3D Gaussian Splatting (3DGS) [37]. Beyond geometric complexity, scaling 3DGS to large environments introduces further difficulties. When generalizing to large scenes, methods face issues such as slow convergence and training instability, which can hinder parallel training efficiency and lead to suboptimal reconstruction quality [12].

Extending 3DGS to effectively represent and render dynamic scenes is another significant challenge. Current implementations exhibit limited support for dynamic objects [18]. Accurately modeling complex non-rigid motion over time poses substantial difficulties, requiring robust techniques to track, deform, and render the scene elements consistently across frames [14,18,29,33]. Addressing these limitations is crucial for enabling 3DGS to handle real-world scenarios involving movement and change.

7.3 Efficiency, Memory, and Scalability

The inherent structure of 3D Gaussian Splatting (3DGS) models, which represent a scene using a large number of explicit 3D Gaussians, presents significant challenges regarding efficiency, memory consumption, and scalability, particularly for extensive environments. A notable concern is the storage requirement; 3DGS models typically necessitate gigabytes (GBs) of storage, posing a limitation when compared to Neural Radiance Fields (NeRF) models, which often require only megabytes (MBs) [7]. This substantial memory footprint becomes particularly pronounced when rendering very large scenes, which still demand significant memory resources [4].

These memory demands contribute directly to scalability limitations. Reconstructing and rendering large-scale complex scenes efficiently while avoiding excessive memory usage is a key technical hurdle [2,14,24]. Research efforts are directed towards mitigating these issues. For instance, approaches like CityGaussianV2 have been developed to address the challenges of generalizing 2DGS techniques to large-scale scenarios, demonstrating effectiveness in avoiding memory explosion problems and achieving notable memory optimization compared to earlier iterations [12].

Beyond static storage and rendering memory, the computational complexity—specifically during the training process—also contributes to efficiency and scalability challenges. Increased training time is one aspect of this complexity. Furthermore, vulnerabilities, such as the Poison-Splat attack, highlight potential weaknesses that can exacerbate these issues by increasing GPU memory usage and training time, potentially leading to system overload [9]. While the provided digests primarily emphasize memory aspects, particularly for large scenes and storage size compared to NeRF, these factors inherently impact the feasibility of achieving real-time performance, especially on hardware with limited computational and memory resources [14].

7.4 Data Efficiency and Generalization

Three-dimensional Gaussian Splatting (3DGS) techniques fundamentally rely on multi-view images as input for scene reconstruction [18]. This requirement inherently introduces challenges related to data efficiency and generalization, particularly concerning data sparsity [2]. The quality of the reconstructed 3D scene is significantly impacted by the density and distribution of the input views. When the number of views is limited or the viewpoints are sparse, it becomes difficult to accurately represent areas of the scene that are not well-covered by the input images. This data sparsity can lead to artifacts, incomplete reconstructions, or reduced fidelity in undersampled regions. Consequently, achieving data-efficient training – training robust models with fewer input views – remains a critical challenge in 3DGS [2]. Furthermore, ensuring that models trained on specific datasets generalize well to new scenes or different viewing conditions also presents difficulties under limited data scenarios [2]. Addressing these issues is crucial for expanding the applicability of 3DGS in scenarios where obtaining dense multi-view datasets is impractical or impossible.

7.5 Rendering Quality and Artifacts

Rendering quality remains a significant challenge in 3D Gaussian Splatting (3DGS), with various artifacts potentially impacting visual fidelity and realism. Among these, issues related to depth ordering and alpha blending of Gaussians are particularly notable and can lead to visual inaccuracies [2].

The correct rendering of transparent or overlapping elements heavily relies on the precise sorting of Gaussians along the viewing ray, and inaccuracies in this process can manifest as visible artifacts, degrading the quality of the rendered image.

Consequently, there is a continuous need for the development of more advanced rendering algorithms specifically designed to mitigate these artifacts and enhance the overall realism of 3DGS representations [2].

Beyond depth sorting, other artifacts such as aliasing can also detract from the visual experience. Techniques are being explored to address these issues. For instance, Mip-Splatting has been introduced as a method to combat aliasing artifacts [10]. This approach incorporates a 3D smoothing filter designed to constrain the size of 3D Gaussian primitives based on the maximum sampling frequency determined by the input view [10].

Furthermore, it replaces the conventional 2D expansion filter with a 2D Mip filter, which effectively simulates the behavior of a 2D box filter, contributing to the reduction of aliasing [10].

While Mip-Splatting specifically targets aliasing, addressing the full spectrum of rendering artifacts, including the critical depth sorting challenges, necessitates ongoing research into sophisticated rendering techniques capable of ensuring accurate and visually pleasing reconstructions. The pursuit of high-fidelity 3DGS representations depends significantly on overcoming these rendering-related limitations.

7.6 Security Concerns

The increasing adoption of 3D Gaussian Splatting (3DGS) for scene representation and rendering highlights the importance of addressing potential security vulnerabilities. While advancements primarily focus on efficiency and quality, the susceptibility of 3DGS pipelines to malicious attacks represents a significant concern, particularly regarding computational resources and data integrity.

A notable vulnerability is the Poison-Splat attack, identified as a computation cost attack specifically targeting the 3DGS training process [9]. This attack is executed by strategically perturbing the input images used for 3DGS training [9]. The adversary's goal is to maximize the computational resources required by the 3DGS system to process these poisoned inputs, particularly increasing training costs [9]. The attack can be formally modeled as a max-min bi-level optimization problem, where the inner optimization layer represents the 3DGS algorithm attempting to reconstruct the scene, and the outer layer embodies the attacker seeking optimal image perturbations to escalate resource consumption [9]. The impact of such an attack can be substantial, potentially leading to excessive consumption of system memory (e.g., GPU VRAM) during training, which could overload hardware and potentially cause service disruptions or server downtime, as implied by the concerns raised regarding memory spikes [9]. Attackers can employ variations, including constrained or unconstrained perturbations, with the latter potentially causing more severe damage but being more easily detectable [9].

While some efforts address security aspects of 3DGS assets, such as protecting copyright, integrity, and privacy through techniques like steganography as proposed by GS-Hider [13], these methods focus on embedding information within the point cloud files after training. Such existing measures do not directly mitigate vulnerabilities related to the training process itself or defend against input data poisoning attacks like Poison-Splat that exploit the resource-intensive nature of 3DGS optimization. This gap highlights the limitations of current security approaches in addressing all facets of 3DGS security. Consequently, there is a clear and present need for the development of robust defense mechanisms specifically designed to detect and counteract computation cost attacks and other forms of data poisoning during the critical training phase of 3DGS, ensuring the stability and reliability of 3DGS-based systems.

7.7 Open Problems and Future Research Avenues

Despite its rapid advancements and impressive performance in novel view synthesis, the field of 3D Gaussian Splatting (3DGS) still faces several significant open problems that require further research to fully unlock its potential [25]. Addressing these limitations is crucial for advancing the state of the art and expanding the practical applications of 3DGS.

One key challenge lies in the **data efficiency** of 3DGS methods. Current approaches often require a substantial number of input views for high-quality scene reconstruction and new view generation [5]. Generating accurate 3DGS representations from a limited number of samples (few-shot) remains an active research area [5]. Furthermore, the method can produce artifacts in areas with insufficient observations [5]. Potential future directions to mitigate this include introducing additional information such as depth, utilizing dense probability distributions, exploring pixel-to-Gaussian mapping, and developing techniques for data interpolation or integration in sparsely observed regions [5].

Another critical limitation is **storage efficiency and scalability**, particularly when handling large-scale environments [5,6]. Unlike some other representations (e.g., NeRF, which stores learned MLP parameters), 3DGS models can require considerable storage [5]. Poor scalability necessitates optimizations for both the training phase and the final model size [5].

Future research should focus on developing more efficient data structures and advanced compression techniques, such as Light-Gaussian, to improve storage utilization [5].

Handling **dynamic scenes** also presents a significant challenge for current 3DGS techniques [6]. Extending the method to accurately represent and render scenes with moving objects or changes over time is a necessary step for broader applicability.

Robustness is another area requiring improvement. This includes enhancing resilience to noise and occlusions in the input data [6]. Furthermore, a critical security vulnerability related to data poisoning attacks has been identified, where malicious data can cause excessive memory allocation and potential server crashes [9]. Simple defenses like limiting the number of Gaussians are insufficient as they degrade reconstruction accuracy [9]. Designing more intelligent and effective defenses against such vulnerabilities is a pressing open problem [9].

Enhancing **editing capabilities** is a crucial direction for enabling creative workflows and practical applications [4,14]. Future developments are expected to focus on intelligent editing tools and techniques that allow users to easily modify 3DGS scenes [14].

Addressing the limitations in handling **complex lighting effects**, such as High Dynamic Range (HDR) scenes, is also important [11]. Improving the quality of reconstructed HDR scenes is a key area for future investigation [11].

Other promising future research directions include improving the **generalization ability** of 3DGS models to novel scenes or conditions [14], fostering **integration with other technologies** [14], exploring **cross-modal learning** [14], and achieving better **integration with existing workflows** [4]. Ultimately, the goal is to continue lowering the technical barriers for creators, enabling them to focus on creativity and produce more diverse and rich digital content [22]. Expanding the applications of 3DGS in areas such as Augmented Reality (AR), Virtual Reality (VR), and content creation also represents a significant future avenue [11]. Overall, continued research is anticipated to bring improvements in realism and broader applicability [4].

8. Conclusion

3D Gaussian Splatting (3DGS) represents a significant advancement in the domain of real-time 3D reconstruction and novel view synthesis, fundamentally altering the landscape of scene representation and rendering [7,10,17,20,23]. Originating from traditional point-based rendering methods [29,33], 3DGS employs an explicit representation using anisotropic 3D Gaussian functions projected onto the image plane [1,15,20], effectively bridging the gap between explicit and implicit scene modeling [31].

The core advantages of 3DGS lie in its exceptional efficiency and rendering speed, often cited as orders of magnitude faster than previous neural radiance field (NeRF) based methods while maintaining comparable or state-of-the-art visual quality [11,16,18,19,20,26]. This performance is attributed to key components such as differentiable splatting and adaptive density control [20], alongside competitive training times [27,31]. Further developments have extended its capabilities, enabling robust large-scale scene reconstruction [12], large-scale geometric deformation modeling [8], high-quality HDR scene rendering [11], optimal 2D-to-3D segmentation [6], and even novel applications like steganography [13]. Its memory efficiency relative to voxel-based methods and support for post-processing adjustments further enhance its practicality [18].

The rapid progress in 3DGS technology holds significant potential for broad applications across computer graphics and computer vision [10,14,17,25,29,33,34]. Its real-time capabilities are poised to impact fields such as virtual and augmented reality, computational photography, robotic navigation, digital cultural heritage preservation, film production special effects [14], Simultaneous Localization and Mapping (SLAM), and autonomous driving [21]. The explicit structure and fast rendering facilitate direct 3D reconstruction and editing applications [29,33]. Increased accessibility through user-friendly interfaces and cloud processing is expected to fuel further innovation [22].

Despite its remarkable advantages and broad prospects [14], 3DGS is not without limitations. Challenges persist in handling dynamic scenes effectively [27], managing potentially high memory and disk space usage [27], and ensuring compatibility with existing rendering pipelines [27]. Issues like "broken" Gaussians (overly large, elongated, or redundant) can affect reconstruction quality [27]. Furthermore, the technology is susceptible to vulnerabilities such as data poisoning attacks that can increase computational load [9]. Addressing these challenges, alongside improving data efficiency and optimizing algorithm design, remains crucial for advancing the field [24]. Continued research and development are essential to overcome current limitations and fully unlock the potential of 3DGS, propelling the fields of computer graphics and computer vision forward [2,10,17,32,34].

References

[1] 3D Gaussian Splatting：3D 渲染新纪元 https://blog.csdn.net/2302_80643506/article/details/145718251

[2] 3D Gaussian Splatting 综述：技术、挑战与机遇 https://blog.csdn.net/m0_74310646/article/details/140939423

[3] 3D Gaussian Splatting (3DGS) 三维重建算法入门详解 https://blog.csdn.net/weixin_48978134/article/details/141218932

[4] 3D Gaussian Splatting: A Breakthrough in Realistic https://www.chaos.com/blog/3d-gaussian-splatting-new-frontier-in-rendering

[5] 3D Gaussian Splatting技术：影响与未来 https://it.sohu.com/a/810946348_121124366

[6] FlashSplat: 最优解2D到3D高斯分割，提速50倍 https://cloud.tencent.com/developer/article/2482781

[7] 3D高斯泼溅 (3DGS)：从新视角合成到实时渲染 https://segmentfault.com/a/1190000045168206

[8] 计图开源：高斯网—实时大尺度几何变形的新型高斯溅射方法 https://cg.cs.tsinghua.edu.cn/jittor/news/2024-09-30-11-43-00-00-GaussianMesh/

[9] 3D高斯泼溅算法现漏洞：数据投毒致显存暴涨，或致服务器宕机 https://segmentfault.com/a/1190000046499754

[10] 3D高斯Gaussian Splatting最新进展与应用解析 https://www.bilibili.com/read/cv29185173

[11] HDR-GS：首个可渲染HDR场景的3D高斯模型，速度提升1000倍 https://baijiahao.baidu.com/s?id=1818040753407886776&wfr=spider&for=pc

[12] CityGaussianV2：中科院提出，解决大规模场景三维重建显存爆炸难题 https://baijiahao.baidu.com/s?id=1818307733311392387&wfr=spider&for=pc

[13] GS-Hider: 3D Gaussian Splatting Steganography for http://www.paperreading.club/page?id=228979

[14] 3D高斯溅射：原理、应用与未来 https://ai-bot.cn/what-is-3d-gaussian-splatting/

[15] 3D Gaussian Splatting 学习记录：基于CUDA加速的点云实时渲染 https://blog.csdn.net/weixin_46933478/article/details/134195335

[16] Real-Time Radiance Field Rendering with 3D Gaussia https://paperswithcode.com/paper/3d-gaussian-splatting-for-real-time-radiance

[17] 3D Gaussian Splatting：原理、应用与最新进展 https://www.bilibili.com/read/mobile?id=29057831

[18] 高斯溅射技术：原理、对比、优劣势与应用 https://blog.csdn.net/zhu_zhu_xia/article/details/146535180

[19] 3D Gaussian Splatting 技术详解：快速高质量的新视图合成 https://blog.csdn.net/m0_37604894/article/details/137864723

[20] 3D Gaussian Splatting：实时神经场渲染的实现与解析 https://blog.csdn.net/weixin_44321764/article/details/145633524

[21] 3D Gaussian Splatting 调研：SLAM与自动驾驶应用 https://blog.csdn.net/gwplovekimi/article/details/135397265

[22] 照片生成3D场景：3DGS技术解析与应用 https://www.find.org.tw/index/tech_obser/browse/7eaa3e8ffc7876e74509befd5ed50b8a/

[23] 3D Gaussian Splatting：介绍、基础知识及进展 https://roll.sohu.com/a/761286340_121124366

[24] 3D高斯喷洒(3D Gaussian Splatting)最新综述 https://www.bilibili.com/read/cv30024926

[25] 3D Gaussian Splatting 综述 https://paper.nweon.com/15149

[26] 3D高斯溅射技术：2024科研热点与实时渲染新突破 https://baijiahao.baidu.com/s?id=1790223574185150235&wfr=spider&for=pc

[27] 3D Gaussian Splatting 技术综述：影响与未来 http://mt.sohu.com/a/810946348_121124366

[28] 3D高斯Gaussian Splatting爆火：最新研究进展与应用 https://www.bilibili.com/read/cv29185173/

[29] 3D Gaussian Splatting 最新综述：技术、应用与未来 https://news.sohu.com/a/770105468_121124366

[30] 3D Gaussian Splatting：最新技术详解与SLAM应用 https://learning.sohu.com/a/750306807_121124366

[31] Gaussian Splatting: A Review of 3D Reconstruction https://blog.csdn.net/weixin_44478317/article/details/143615549

[32] 3D Gaussian Splatting 论文及代码汇总（持续更新） https://blog.csdn.net/john_bh/article/details/136981310

[33] 3D高斯溅射(3DGS)最新综述：重建、编辑与应用 https://it.sohu.com/a/770105468_121124366

[34] CVPR2024：3D Gaussian Splatting赋能3D视觉任务革新 https://mp.weixin.qq.com/s?__biz=MzU2NjU3OTc5NA==&mid=2247585261&idx=2&sn=7d660281491ec5ee179e23ed852a10b7&chksm=fda3981c396fe517d2b17adb565218aca7a2ed4f0b2330811ed2d802442289d7813fc318511b&scene=27

[35] 3D高斯溅射 (3DGS) 与 NeRF：异同比较 https://blog.csdn.net/weixin_43890835/article/details/135493398

[36] 3D Gaussian Splatting：原理、优势与快速渲染 https://mp.weixin.qq.com/s?__biz=MzU1NjEwMTY0Mw==&mid=2247589043&idx=1&sn=30b85681589e7b3a750c12ad428af77e&chksm=fa4bca88209fa312385bec3be094bd519cd238f8efd762d9c76d69bb9761432cd70908288cca&scene=27

[37] 3D Gaussian Splatting：原理、优势与应用 https://mp.weixin.qq.com/s?__biz=MzU1NjEwMTY0Mw==&mid=2247593665&idx=1&sn=8bfc3d886981bcfb5a69f77f8a18882b&chksm=fad12a78cdae62beb5ef09f27348c47062c722b367ec3cda9e75e2e4ae1513312833ba133131&scene=27

[38] Gaussian Splatting for 3D Reconstruction and Novel https://blog.csdn.net/weixin_44478317/article/details/143596502