From Volume Rendering to 3D Gaussian Splatting: Theory and Applications

Vitor Pereira Matias[*§], Daniel Perazzo[*†], Vinicius Silva[‡], Alberto Raposo[‡],
Luiz Velho[†], Afonso Paiva[§], and Tiago Novello[†]
ICMC-USP[§], IMPA[†], PUC-RIO[‡]

***Abstract*—The problem of 3D reconstruction from posed images is undergoing a fundamental transformation, driven by continuous advances in 3D Gaussian Splatting (3DGS). By modeling scenes explicitly as collections of 3D Gaussians, 3DGS enables efficient rasterization through volumetric splatting, offering thus a seamless integration with common graphics pipelines. Despite its real-time rendering capabilities for novel view synthesis, 3DGS suffers from a high memory footprint, the tendency to bake lighting effects directly into its representation, and limited support for secondary-ray effects. This tutorial provides a concise yet comprehensive overview of the 3DGS pipeline, starting from its splatting formulation and then exploring the main efforts in addressing its limitations. Finally, we survey a range of applications that leverage 3DGS for surface reconstruction, avatar modeling, animation, and content generation—highlighting its efficient rendering and suitability for feed-forward pipelines.**

***Index Terms*—Gaussian Splatting, Volume Rendering, 3D Reconstruction.**

I. Introduction

3D reconstruction from posed images is a long-standing problem in visual computing that is undergoing a fundamental disruption, driven by advances in Neural Radiance Fields (NeRFs) [1] and 3D Gaussian Splatting (3DGS) [2]. Given a set of input views with known poses from an unknown scene, the goal is to optimize the parameters of a 3D representation that accurately captures the scene's geometry and appearance. NeRFs have had a significant impact on this task by representing scene geometry (as volume density) and radiance using neural networks, optimized through differentiable volume rendering [3]. This framework enables highly detailed Novel View Synthesis (NVS) of real-world scenes, with straightforward integration of 3D reconstruction into deep learning pipelines.

However, representing a scene volumetrically through a neural network can be inefficient, as the network must learn to represent both occupied and empty regions of the domain. During training, its global representation requires supervision across the entire domain, including empty space, resulting in high computational costs and making real-time rendering impractical–crucial for applications. To address these challenges, Kerbl et al. [2] introduced 3DGS, which models the scene as a collection of 3D Gaussians with colors and leverages volume splatting [4] for differentiable rendering. By avoiding costly queries in empty space and employing rasterization, 3DGS achieves both high detail and real-time performance. Figure 1 gives an overview of 3DGS.

[*]denotes equal contribution.

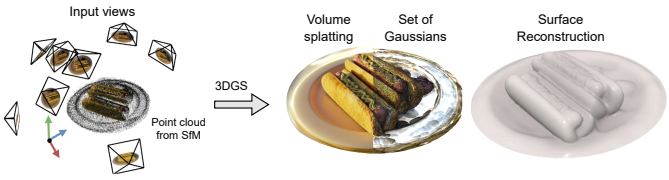

Fig. 1: Illustration of the 3DGS pipeline. Given a set of posed input images (left), a sparse point cloud from Structure-from-Motion (SfM) is used to initialize a set of colored 3D Gaussians (middle). These Gaussians are then optimized via volume splatting and can support downstream tasks such as novel view synthesis and surface reconstruction (right).

The objective of this tutorial is to present the 3DGS method by deriving its splatting formulation from the volume rendering equation, along with techniques for Gaussian initialization and adaptation during training. We then review recent approaches that address key limitations of the original 3DGS method, including its high memory consumption and limited support for secondary-ray effects. Finally, we discuss a range of applications of 3DGS, including surface reconstruction, animation, avatar modeling, and feed-forward 3D reconstruction from sparse views. In summary, our contributions are:

- An intuitive mathematical derivation of 3DGS from the volume rendering equation.
- A survey of 3DGS extensions and applications across a variety of 3D reconstruction tasks.

II. Volume Rendering Preliminaries

Our objective is to use differentiable rendering to reconstruct a 3D scene—parameterized by θ—from a set of N posed multi-view images and their corresponding camera intrinsics and extrinsics $\{\mathbf{I}_i, \mathbf{K}_i, \mathbf{W}_i\}_{i=1}^{N}$. For each view i, we render the scene using the current θ, obtaining an image $\mathbf{I}_{\theta,i}$, which is then compared to its corresponding input view $\mathbf{I}_i$, enforcing $\mathbf{I}_i \approx \mathbf{I}_{\theta,i}$. This is achieved by minimizing the photometric loss:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \left\|\mathbf{I}_{\theta,i} - \mathbf{I}_i\right\|^2, \tag{1}$$

using standard gradient descent algorithms such as Adam [5].

Traditional mesh-based representations are typically non-differentiable, making them unsuitable for optimizing loss (1) via gradient descent methods. To overcome this, NeRFs [1] and 3DGS [2] employed *volumetric representations* to enable differentiable volume rendering. Specifically, the scene is parameterized by a density field $\sigma_\theta : \mathbb{R}^3 \to \mathbb{R}$, quantifying

particle light absorption, and a color field $c_\theta : \mathbb{R}^3 \to \mathbb{R}$, quantifying emitted radiance. Thus, treating the scene as a "cloud" that both absorbs and emits light. Although seemingly unsuitable for representing solid objects initially, this approach has yielded strong results in novel view synthesis [6], [7] and detailed surface reconstruction [8], [9].

Let $\gamma(t) = \mathbf{o_i} + t\mathbf{v}$ be the view ray associated with a pixel p in the input image $\mathbf{I}_i$, where $\mathbf{o_i}$ is the camera position. Our goal is to compute the color observed along this ray using volume rendering techniques [10]. We restrict the ray domain to the interval $[t_n, t_f]$, where t_n and t_f correspond to the near and far bounds. Ignoring scattering, the accumulated radiance $I(\gamma(t))$ along the ray satisfies the volumetric light transport ordinary differential equation (ODE) [3]:

$$I'(t) = \underbrace{\sigma_\theta(t)c_\theta(t)}_{\text{Emission}} - \underbrace{\sigma_\theta(t)I(t)}_{\text{Absorption}}. \tag{2}$$

For simplicity, we omit γ in the notation. The ODE (2) can be solved by separation of variables [3, Sec. 4]. Assuming zero background emission as the initial condition, i.e., $I(t_n) = 0$, the accumulated radiance at the end of the ray is given by:

$$I_f := I(t_f) = \int_{t_n}^{t_f} \sigma_\theta(t)c_\theta(t) \exp\left(-\int_{t_n}^{t} \sigma_\theta(s)\, ds\right) dt. \tag{3}$$

The *transmittance* $\exp(\cdots)$ captures the attenuation of light due to absorption along the ray. In practice, to render the predicted image $\mathbf{I}_{\theta,i}$ for view i, we approximate the integral (3) using numerical quadrature methods resulting in the volume rendering equation [1]:

$$I_f \approx \sum_{i=1}^{N} c_\theta(t_i)\Big(1-\exp\big(-\sigma_\theta(t_i)\delta_i\big)\Big)\prod_{j=1}^{i-1}\exp\big(-\sigma_\theta(t_j)\delta_j\big), \tag{4}$$

where $\delta_i = t_i - t_{i-1}$ is the step size between consecutive samples. This formulation allows gradient-based optimization, as it is fully differentiable. In NeRF, the functions σ_θ and c_θ are modeled by neural networks, and (4) is used to optimize θ via the photometric loss (1). However, the global nature of neural representations requires supervision across the entire domain—including empty space—to compute (4). This results in high computational overhead and severely limits real-time rendering capabilities, essential for applications.

III. 3D Gaussian Splatting

Gaussian splatting overview. To overcome the computational cost of evaluating the volume rendering equation (3), Kerbl et al. [2] introduced 3D Gaussian Splatting (3DGS), which represents the density and radiance fields σ_θ and c_θ using a collection of colored 3D Gaussians, rendered efficiently via volume splatting [4]. We follow the pipeline illustrated in Fig. 2 to describe the main stages of 3DGS. The input consists of a set of posed images $\{\mathbf{I}_i\}$, from which a colored point cloud is obtained via Structure-from-Motion (SfM) [11]; see Fig. 2 (top-left). This point cloud serves as the basis for initializing a set of M Gaussians, $g_i := \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \sigma_i, c_i\}$, where $\boldsymbol{\mu}_i \in \mathbb{R}^3$ is the Gaussian center, $\boldsymbol{\Sigma}_i \in \mathbb{R}^{3\times3}$ is the covariance matrix, $\sigma_i \in \mathbb{R}$ is the opacity, and $c_i \in \mathbb{R}^3$ is the RGB color. Each Gaussian defines a density function $\mathcal{G}_i : \mathbb{R}^3 \to \mathbb{R}$ given by:

$$\mathcal{G}_i(\mathbf{x}) := \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right), \tag{5}$$

which controls the spatial influence of the Gaussian around point $\mathbf{x}$. To avoid expensive evaluation in empty space, the initial Gaussians can be placed where geometry is present, i.e. the optimization begins with the Gaussian centers $\boldsymbol{\mu}_i$ and colors c_i initialized from the SfM point cloud; see Fig. 2 (top-middle). The scene is then rendered via *volume splatting*, and the output image $\mathbf{I}_{\theta,i}$ is compared to the corresponding input view $\mathbf{I}_i$ using a photometric loss. The resulting error is backpropagated to update the Gaussian parameters via gradient descent. To further refine the representation and avoid poor local minima, an adaptation module is employed during training; see Fig. 2 (bottom-right). This step dynamically adjusts the number of Gaussians by splitting Gaussians that are too large, cloning Gaussians that underfit local detail, and pruning those with persistently low opacity.

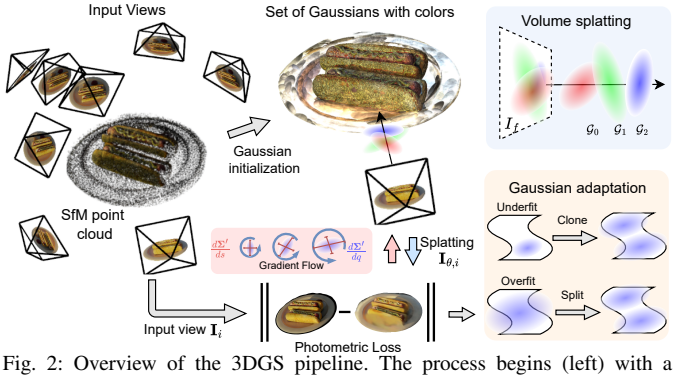

Fig. 2: Overview of the 3DGS pipeline. The process begins (left) with a set of posed images captured around an object, from which a sparse SfM point cloud is reconstructed. Gaussians are then initialized over this point cloud and optimized (center) through differentiable volumetric splatting. The rendered image is compared to the input views using an photometric loss, whose gradient is used to update the Gaussian parameters. To enhance spatial coverage and avoid under- or over-representation, 3DGS incorporates an adaptation step (right) that dynamically adds (via splitting or cloning) or removes Gaussians during training.

Volume splatting. While it is possible to apply quadrature-based volume rendering (4) to the Gaussian representation, this is computationally expensive since it would require querying points outside the Gaussian supports. Instead, 3DGS adopts *volume splatting*—a efficient rasterization-based alternative—to approximate the integral in (3). Precisely, let γ be a view ray associated to a pixel $\mathbf{p}$, intersecting a set of K Gaussians sorted according to the distance from their centers to the camera position. Using basic properties from integral calculus and assuming that each Gaussian has local support, we can rewrite (3) as:

$$I(\gamma) = \sum_{i=1}^{K} c_i\sigma_i\tau_i\prod_{j=1}^{i-1}(1-\sigma_j\tau_j), \text{ with } \tau_i = \int_{\mathbb{R}} \mathcal{G}_i\big(\gamma(t)\big)dt. \tag{6}$$

Zwicker [4] show that τ_k corresponds to projecting (i.e., *splatting*) the Gaussian and evaluating it at the pixel location $\mathbf{p}$.

To show that $\int_{\mathbb{R}} \mathcal{G}_i\big(\gamma(t)\big)\, dt$ results in a 2D Gaussian in the image plane, we assume that coordinates are aligned with the camera coordinate system defined by the pose matrix $\mathbf{W}$. Let $P_{\mathbf{K}} : \mathbb{R}^3 \to \mathbb{R}^2$ denote the perspective projection induced by the camera intrinsics $\mathbf{K}$. Since $P_{\mathbf{K}}$ is non-affine, Zwicker et al. [4] proposed a first-order approximation of $P_{\mathbf{K}}$ around the Gaussian center $\boldsymbol{\mu}_i$ using its Jacobian $\mathbf{J} \in \mathbb{R}^{2\times 3}$:

$$P_{\mathbf{K}}(\mathbf{x}) \approx P_{\mathbf{K}}(\boldsymbol{\mu}_i) + \mathbf{J}(\mathbf{x} - \boldsymbol{\mu}_i). \tag{7}$$

This approximation allows computing the integral τ_i by evaluating a 2D Gaussian $\tilde{\mathcal{G}}_i : \mathbb{R}^2 \to \mathbb{R}$, called *splatting* of the 3D Gaussian $\mathcal{G}_i$. To compute the the mean and covariance of this Gaussian, we express the 3D Gaussian into the camera coordinate system using $\mathbf{W}$ and apply the linearized projection using our first-order approximation:

$$\tilde{\boldsymbol{\mu}}_i := P_{\mathbf{K}}(\mathbf{W}\boldsymbol{\mu}_i), \qquad \tilde{\boldsymbol{\Sigma}}_i := \mathbf{J}\mathbf{W}\boldsymbol{\Sigma}_i\mathbf{W}^T\mathbf{J}^T. \tag{8}$$

The opacity and colors are preserved, that is, $\tilde{\sigma}_i = \sigma_i$ and $\tilde{c}_i = c_i$. Thus, the resulting *Gaussian splatting* set is given by $\tilde{g}_i = (\tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\Sigma}}_i, \tilde{\sigma}_i, \tilde{c}_i,)$ is called. Now, we must perform the composition of multiple volumes with different opacities (alpha-compositing) with the projected Gaussians. Assuming the K Gaussians intersecting the ray "shooting" from pixel $\boldsymbol{p} \in \mathbb{R}^2$ are sorted by depth, the final light intensity I_f at that pixel $\boldsymbol{p}$ is given by:

$$I_f = \sum_{i=1}^{N} c_i\, \tilde{\sigma}_i\, \tilde{\mathcal{G}}_i(\boldsymbol{p}) \prod_{j=1}^{i-1} \Big(1 - \tilde{\sigma}_j\, \tilde{\mathcal{G}}_j(\boldsymbol{p})\Big). \tag{9}$$

With the new intensity equation, the pipeline is reformulated to first project (i.e., splat) the Gaussians onto the image, followed by pixel evaluation. A tile-based *rasterization* strategy enables parallelization for faster volume rendering. Additionally, to reduce aliasing artifacts, 3DGS uses a dilation-based filter [12]. It is important to note that, since both NeRFs and Gaussian Splatting are derived from the volume rendering integral (3), note that volume splatting (9) resembles volume rendering (4). Figure 2 provides an overview of the volume splatting operation, where Gaussians are first sorted along the viewing ray and then alpha-composited to form the final image.

Color representation. In 3DGS [2], this modeling approach is used to perform 3D reconstruction from posed images, where the scene parameters θ are defined as the set of Gaussians. They modeled the colors using spherical harmonics to represent view angle light variation and used a diagonalization trick to parameterize the covariance matrix. Since $\boldsymbol{\Sigma}_i$ is a positive-definite matrix, hence, there is a diagonalization such that $\boldsymbol{\Sigma}_i := \mathbf{V}\mathbf{S}\mathbf{V}^T$, where $\mathbf{S} \in \mathbb{R}^{3\times 3}$ is a diagonal matrix with positive entries and $\mathbf{V} \in SO(3)$, where $SO(3)$ is the 3D rotation group. With this representation, we can optimize $\mathbf{S}$ by directly optimizing its diagonal entries, represented as a vector $\mathbf{s} \in \mathbb{R}^3$. For $\mathbf{V}$, the rotation can be parameterized as an optimizable quaternion $\mathbf{q} \in \mathbb{R}^4$.

IV. Extensions and Developments

The original 3D Gaussian Splatting (3DGS) technique achieves high-quality view synthesis but faces challenges such as memory usage and accurate volume rendering. Recent extensions address these issues through various improvements. Thanks to its strong performance, 3DGS is also being applied beyond its original scope, including surface reconstruction and avatar creation. In this section, we review key developments and highlight notable contributions.

Memory: While possessing high quality, 3DGS employs a substantial quantity of Gaussians, approximately 200,000–500,000 for complex scenes, thereby resulting in significant memory and storage demands, particularly when contrasted with the requirements of NeRFs. Furthermore, these challenges can be exacerbated by the adaptation mechanism, which may introduce additional Gaussians during the training process. Consequently, certain methodologies [13]–[15] have implemented running regimes during the inclusion process to reduce the number of Gaussians, thus minimizing the memory footprint. Recent studies have also endeavored to refine the adaptation strategy, exemplified by MCMC-3DGS [16], wherein the traditional heuristics of Gaussian adaptation (densify and prune) have been supplanted by a Markov Chain Monte Carlo technique utilizing a loss function.

Aliasing and multi-resolution: 3DGS uses a dilation filter, which fails to suppress high-frequency artifacts when varying focal length or camera distance. To overcome this, MIP-Splatting [17] introduced both 2D and 3D Gaussian filters, improving robustness to aliasing caused by resolution changes. Building on this, recent work has explored Gaussian Splatting for multiresolution applications [18].

Specularity: In 3DGS, an emission-absorption model (2) is used instead of a scattering model [3], effectively baking lighting conditions into the spherical harmonics coefficients. This limits performance on highly reflective surfaces and prevents relighting. To address this, methods such as GaussianShader [19], 3DGS-DR [20], and IRGS [21] incorporate classical reflection and shading concepts into the 3DGS framework. An alternative approach by Ginter et al. [22] uses diffusion models for relighting. Gao et al. [23] embed BRDF parameters—albedo, roughness, surface normals, and incident lighting—into each Gaussian, enabling per-point physically based rendering. Direct lighting is reconstructed using environment maps, while spherical harmonics model indirect lighting. Additionally, while standard 3DGS relies only on primary rays, recent methods incorporate secondary rays to model interreflections. To this end, ray tracing techniques have been integrated [21], [24], allowing secondary rays—such as reflections and refractions—to be spawned upon primary ray interactions with Gaussians.

Volume Splatting Revisited: Volume splatting (9) introduces several approximations that can compromise rendering accuracy. Stop-the-pop [25] mitigates popping artifacts caused by the sorting procedure. Other works [26], [27] propose

compensation terms on the opacity for more accurate volume rendering. Additionally, to improve the affine approximation of the projection operator (7), recent methods such as 3DGUT [28] employ an unscented transform for more precise Gaussian projection onto the image plane.

3D Reconstruction In the Wild: Recent works have extended 3DGS to "in-the-wild" settings characterized by occlusions, transient objects, and varying illumination. WildGaussians [29] integrates robust DINO features and a lightweight appearance modeling module within the 3DGS framework to handle unconstrained photo collections, matching real-time rendering speeds while outperforming the original NeRF baseline and having similar speed to 3DGS on wild data. Gaussian in the Wild (GS-W) [30] separates intrinsic and dynamic appearance attributes per Gaussian point, employs an adaptive sampling strategy, and leverages 2D visibility maps to mitigate transient occlusions and photometric variations, achieving high-fidelity reconstructions at fast render times. In a similar approach, Wild-GS [31] aligns pixel appearance features from image triplanes directly to 3D Gaussians and uses depth regularization and visibility maps to mitigate the transient effects and constrain the geometry. SWAG [32] extends 3DGS by conditioning Gaussians on learned appearance embeddings and training unsupervised transient Gaussians to ignore occluders, yielding state-of-the-art results on diverse photo scenes. SpotlessSplats [33] further enhances robustness by detecting outliers in a richer, pre-trained feature space to ignore transient distractors, improving reconstruction quality in casual captures.

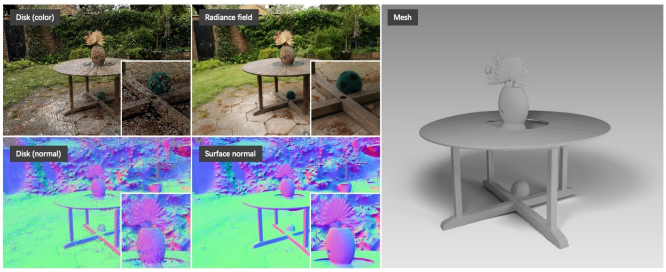

Fig. 3: Illustration of 2DGS [9]. The method represents the object as a set of 2D disks and successfully recovers both high-quality view synthesis and high-resolution normal maps. Image from Huang et al. [9].

Surface reconstruction: Although 3DGS has shown strong performance in NVS, it is not directly suited for mesh extraction. To address this limitation, several techniques have been proposed to augment 3D Gaussian Splatting with surface reconstruction capabilities. One common direction aims to improve the spatial localization of the Gaussians. For instance "flattening" the 3D Gaussians [34], enables more accurate localization which was further extended in 2DGS [9], replacing 3D Gaussians with 2D, as shown in Figure 3. Leveraging neural signed distance functions (SDFs) for surface reconstruction — inspired by similar approaches used in NeRF-based methods [8], [35] — techniques such as GSDF [36] and GSPull [37] adopt this strategy to recover more detailed mesh structures. Additionally, some recent approaches have proposed rethinking the splatting operation [38]. Several recent approaches have been proposed to address this problem more efficiently, including methods that enable significantly faster reconstruction [39], reconstruction from sparse input views [40], [41], and techniques specialized for urban driving scenes [42].

Animation: 3D representations are often used to enable animation or to recreate dynamic behavior. To achieve physically accurate dynamics, PhysGaussian [43] integrates physics-based continuum mechanics directly within 3DGS, creating a unified simulation-to-rendering pipeline in which each Gaussian is treated as a discrete physical particle with volume, allowing direct simulation. Furthermore, Gaussian Splashing [44] performs physical simulations using Position-Based Fluids. To represent fluids more accurately, it builds upon GaussianShader [19], which enriches Gaussians with material parameters such as diffusiveness, specularity, roughness, and normals to handle specular and reflective objects. To ensure physically accurate simulation, they reconstruct surfaces and disentangle the objects that will be actively simulated, using a mesh-based simulation framework with Gaussians attached to the meshes for texture rendering. Additionally, 4D-GS [45] proposes a method for modeling dynamic Gaussians for time-varying content, such as videos. They develop a spatiotemporal structure encoder to capture deformations of both positions and colors over time.

Avatars: Reconstructing human avatars using 3DGS poses challenges, such as handling dynamic Gaussians and supporting relighting. To tackle these issues, Gaussian Avatars [46] trains a FLAME model [47] of the bust, associating each triangle with a Gaussian that is optimized for the scene while being constrained to remain within its corresponding triangle. In contrast, Gaussian Head Avatar (GHA) [48] uses two MLPs and a deformation module to predict the 3D Gaussian positions based on facial expression and head pose, alongside a color MLP. Gaussian Avatar [49] leverages the SMPL model [50] as a prior to reconstruct full-body avatars using pose features to predict Gaussian parameters through a feed-forward network. Similarly, 3DGS-Avatar [51] reconstructs full-body avatars by combining Gaussians with both non-rigid and rigid transformations of an underlying SMPL model. Novel approaches have further advanced avatar generation: GPAvatar [52] achieves high-resolution results, MeGA [53] supports editing capabilities, and DAGSM [54] enables generative avatars from text descriptions. Additionally, Relightable Gaussian Codec Avatars [55] introduced relightable Gaussian avatars, supporting more diverse lighting conditions. Figure 4 summarizes these methods for avatar creation.

3D reconstruction from sparse views: 3DGS [2] often struggles with NVS from sparse image sets, as its optimization can become trapped in local minima [56], [57]. To address this challenge, several works have proposed training feed-forward networks (FFNs) to directly predict 3D Gaussian parameters, supervised by photometric losses such as LPIPS,

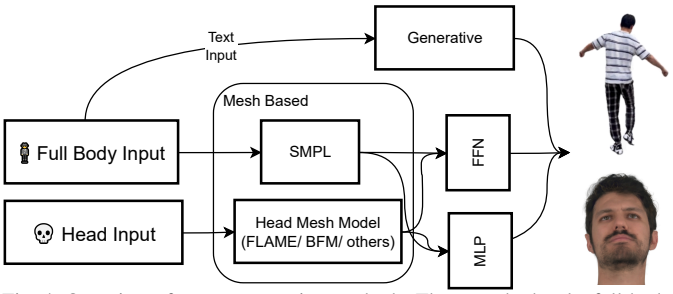

Fig. 4: Overview of avatar generation methods. These methods take full-body or head-only inputs, typically processed using body models like SMPL [50] or FLAME [47], respectively. Some approaches use feed-forward networks (FFNs) to generate Gaussians, while others employ MLPs for color effects. Additionally, some methods use text input to guide avatar generation.

SSIM, or similar metrics, as shown in Figure 2. For instance, Flash3D [58] uses an FFN for monocular reconstruction, generating 3DGS scenes by estimating depth from a single view. PixelSplat [56], incorporates an epipolar transformer to generalize features between two views, pairing this with a probabilistic prediction of Gaussians aligned along camera-to-pixel rays. MVSplat [59] builds upon PixelSplat by combining multi-view feature extraction through transformers and cost volumes, which are then processed by a U-Net. Similarly, GS-LRM [60] tokenizes two to four images and their camera parameters, concatenates them, and passes them through a transformer block whose output tokens are decoded into Gaussian parameters. NoPoSplat [61] addresses critical limitations by eliminating the need for known camera poses. It uses a canonical coordinate system anchored to a single input view and introduces intrinsics token embeddings to resolve scale ambiguity, achieving superior novel view synthesis quality, especially in scenarios with minimal view overlap. Beyond these FFN-based approaches, CoR-GS [57] improves sparse-view reconstruction by defining two independent reconstructions and comparing them to discard inaccurate Gaussians. Additionally, MAtCha [41] employs an atlas-of-charts and depth estimation as input to an MLP, which deforms the atlas and produces high-quality meshes represented with 2D Gaussians.

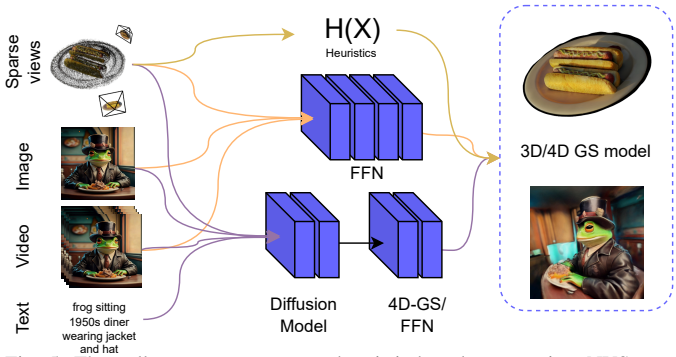

Fig. 5: The yellow arrows represent heuristic-based sparse view NVS, e.g. CoR-GS [57] and MAtCha [41], orange represents FFNs for NVS from sparse views, single image, or video [56], [58]–[61]. Finally, purple shows diffusion models being used to generate content from all sorts of inputs [62]–[68].

Generative models: Image or video diffusion models can aid Gaussian Splating as both input and output enhancers. Additionally, 3D content may also be generated with diffusion models that output a set of 3D Gaussians, and can be expanded to 4D animatable objects. LGM [62], GRM [63], and DreamGaussian [64] use multi-view diffusion models as priors 3D generators from text or image inputs, which are passed into a large FFN model that predicts 3D Gaussians to perform 3D reconstruction. The input and architecture of these models vary, as LGM uses an Asymmetric U-Net taking as inputs the MV images with ray embeddings; GRM uses ViT features from the MV images, which are fed into an Upsampler transformer capable of up-sampling the ViT features into a large number of 3D Gaussians. Additionally, LaRa [65] defines a volume transformer taking as input DINO image features from diffusion models or real data to generate 3D content. Extending to 4D content, Wang et. al. [66] reconstruct a 4D scene from a single monocular in-the-wild video, using the RGB video plus their respective monocular depths and an additional 2D point tracking over time. L4GM [67] also produces 4D scenes from a single monocular video, contrary to Wang et. al. they use diffusion models that takes as input videos and generates multi view videos which are then passed onto an LGM-like network, which results in better overall geometry. CAT4D [68] improves results by creating a multi view video diffusion model from a set of 4D views as input, they use a camera-temporal sampling that allows for separate camera and time controls, and allows for time-spatial consistency across all multi-views videos; these are then inputted into 4D-GS to generate a dynamic 3D scene. Figure 5 shows an overview of these methods.

V. Conclusion and Open Problems

Gaussian Splatting introduced a novel approach to 3D reconstruction, enabling the creation of high-quality representations. As discussed in this survey, there remain many open problems and promising research directions, such as determining the optimal number of Gaussians, improving the splatting formulation itself, and developing feed-forward 3D reconstruction models that are both fast to use and robust to sparse sets containing any number of input views.

Acknowledgments: We would like to thank CAPES, as this study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 (grant 88887.842584/2023-00); CNPq; FAPESP; FAPERJ; and Google for partially funding this work.

References

[1] B. Mildenhall *et al.*, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, 2021.
[2] B. Kerbl *et al.*, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, 2023.
[3] N. Max, "Optical models for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, 1995.
[4] M. Zwicker *et al.*, "Ewa splatting," *IEEE Transactions on Visualization and Computer Graphics*, 2002.
[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv*, 2014.
[6] K. Gao *et al.*, "Nerf: Neural radiance field in 3d vision, a comprehensive review," *arXiv*, 2022.
[7] G. Chen and W. Wang, "A survey on 3d gaussian splatting," *TPAM*, 2024.

[8] P. Wang *et al.*, "Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction," *NeurIPS*, 2021.

[9] B. Huang *et al.*, "2d gaussian splatting for geometrically accurate radiance fields," in *SIGGRAPH*, 2024.

[10] R. A. Drebin *et al.*, "Volume rendering," *SIGGRAPH*, 1988.

[11] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4104–4113.

[12] J. T. Barron *et al.*, "Mip-nerf 360: Unbounded anti-aliased neural radiance fields," in *CVPR*, 2022.

[13] Z. Fan *et al.*, "Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps," *NeurIPS*, vol. 37, pp. 140 138–140 158, 2024.

[14] T. Lu *et al.*, "Scaffold-gs: Structured 3d gaussians for view-adaptive rendering," in *CVPR*, 2024, pp. 20 654–20 664.

[15] M. Niemeyer *et al.*, "Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps," 2025.

[16] S. Kheradmand *et al.*, "3d gaussian splatting as markov chain monte carlo," *NeurIPS*, vol. 37, pp. 80 965–80 986, 2024.

[17] Z. Yu *et al.*, "Mip-splatting: Alias-free 3d gaussian splatting," *CVPR*, 2024.

[18] Z. Yan *et al.*, "Multi-scale 3d gaussian splatting for anti-aliased rendering," in *CVPR*, 2024, pp. 20 923–20 931.

[19] Y. Jiang *et al.*, "Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces," in *CVPR*, 2024.

[20] K. Ye *et al.*, "3d gaussian splatting with deferred reflection," in *SIGGRAPH*, 2024.

[21] C. Gu *et al.*, "Irgs: Inter-reflective gaussian splatting with 2d gaussian ray tracing," in *CVPR*, 2025.

[22] Y. Poirier-Ginter *et al.*, "A diffusion approach to radiance field relighting using multi-illumination synthesis," in *Computer Graphics Forum*, vol. 43, no. 4. Wiley Online Library, 2024, p. e15147.

[23] J. Gao *et al.*, "Relightable 3d gaussians: Realistic point cloud relighting with brdf decomposition and ray tracing," in *ECCV*. Springer, 2024, pp. 73–89.

[24] N. Moenne-Loccoz *et al.*, "3d gaussian ray tracing: Fast tracing of particle scenes," *SIGGRAPH Asia*, 2024.

[25] L. Radl *et al.*, "StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering," *ACM Transactions on Graphics*, vol. 43, no. 4, 2024.

[26] A. Celarek *et al.*, "Does 3d gaussian splatting need accurate volumetric rendering?" in *Computer Graphics Forum*. Wiley Online Library, 2025, p. e70032.

[27] C. Talegaonkar *et al.*, "Volumetrically consistent 3d gaussian rasterization," in *CVPR*, 2025, pp. 10 953–10 963.

[28] Q. Wu *et al.*, "3dgut: Enabling distorted cameras and secondary rays in gaussian splatting," *CVPR*, 2025.

[29] J. Kulhanek *et al.*, "Wildgaussians: 3d gaussian splatting in the wild," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[30] D. Zhang *et al.*, "Gaussian in the wild: 3d gaussian splatting for unconstrained image collections," in *European Conference on Computer Vision*. Springer, 2024, pp. 341–359.

[31] J. Xu *et al.*, "Wild-gs: Real-time novel view synthesis from unconstrained photo collections," *Advances in Neural Information Processing Systems*, vol. 37, pp. 103 334–103 355, 2024.

[32] H. Dahmani *et al.*, "Swag: Splatting in the wild images with appearance-conditioned gaussians," in *European Conference on Computer Vision*. Springer, 2024, pp. 325–340.

[33] S. Sabour *et al.*, "Spotlesssplats: Ignoring distractors in 3d gaussian splatting," *ACM Transactions on Graphics*, vol. 44, no. 2, pp. 1–11, 2025.

[34] A. Guédon and V. Lepetit, "Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering," in *CVPR*, 2024.

[35] Y. Wang *et al.*, "Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction," in *ICCV*, 2023.

[36] M. Yu *et al.*, "Gsdf: 3dgs meets sdf for improved rendering and reconstruction," *Neurips*, 2025.

[37] W. Zhang *et al.*, "Neural signed distance function inference through splatting 3d gaussians pulled on zero-level set," in *NeurIPS*, 2024.

[38] Z. Yu *et al.*, "Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes," *ACM TOG*, 2024.

[39] B. Tan *et al.*, "Planarsplatting: Accurate planar surface reconstruction in 3 minutes," in *CVPR*, 2025, pp. 1190–1199.

[40] J. Wu *et al.*, "Sparse2dgs: Geometry-prioritized gaussian splatting for surface reconstruction from sparse views," in *CVPR*, 2025, pp. 11 307–11 316.

[41] A. Guédon *et al.*, "Matcha gaussians: Atlas of charts for high-quality geometry and photorealism from sparse views," *CVPR*, 2025.

[42] C. Peng *et al.*, "Desire-gs: 4d street gaussians for static-dynamic decomposition and surface reconstruction for urban driving scenes," in *CVPR*, 2025, pp. 6782–6791.

[43] T. Xie *et al.*, "Physgaussian: Physics-integrated 3d gaussians for generative dynamics," in *CVPR*, 2024.

[44] Y. Feng *et al.*, "Gaussian splashing: Unified particles for versatile motion synthesis and rendering," in *CVPR*, 2025.

[45] G. Wu *et al.*, "4d gaussian splatting for real-time dynamic scene rendering," in *CVPR*, 2024.

[46] S. Qian *et al.*, "Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians," in *CVPR*, 2024.

[47] T. Li *et al.*, "Learning a model of facial shape and expression from 4D scans," *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, vol. 36, no. 6, pp. 194:1–194:17, 2017. [Online]. Available: https://doi.org/10.1145/3130800.3130813

[48] Y. Xu *et al.*, "Gaussian head avatar: Ultra high-fidelity head avatar via dynamic gaussians," in *CVPR*, June 2024, pp. 1931–1941.

[49] L. Hu *et al.*, "Gaussianavatar: Towards realistic human avatar modeling from a single video via animatable 3d gaussians," in *CVPR*, 2024.

[50] M. Loper *et al.*, "SMPL: A skinned multi-person linear model," *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, vol. 34, no. 6, pp. 248:1–248:16, Oct. 2015.

[51] Z. Qian *et al.*, "3dgs-avatar: Animatable avatars via deformable 3d gaussian splatting," in *CVPR*, 2024.

[52] W.-Q. Feng *et al.*, "Gpavatar: High-fidelity head avatars by learning efficient gaussian projections," in *CVPR*, 2025.

[53] C. Wang *et al.*, "Mega: Hybrid mesh-gaussian head avatar for high-fidelity rendering and head editing," in *CVPR*, 2025.

[54] J. Zhuang *et al.*, "Dagsm: Disentangled avatar generation with gs-enhanced mesh," in *CVPR*, 2025.

[55] S. Saito *et al.*, "Relightable gaussian codec avatars," in *CVPR*, 2024, pp. 130–141.

[56] D. Charatan *et al.*, "pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction," in *CVPR*, 2024.

[57] J. Zhang *et al.*, "Cor-gs: sparse-view 3d gaussian splatting via co-regularization," in *ECCV*, 2024.

[58] S. Szymanowicz *et al.*, "Flash3d: Feed-forward generalisable 3d scene reconstruction from a single image," *3DV*, 2024.

[59] Y. Chen *et al.*, "Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images," in *ECCV*, 2024.

[60] K. Zhang *et al.*, "Gs-lrm: Large reconstruction model for 3d gaussian splatting," in *ECCV*, 2024.

[61] B. Ye *et al.*, "No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images," *ICLR*, 2025.

[62] J. Tang *et al.*, "Lgm: Large multi-view gaussian model for high-resolution 3d content creation," in *ECCV*, 2024.

[63] Y. Xu *et al.*, "Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation," in *ECCV*, 2024.

[64] J. Tang *et al.*, "Dreamgaussian: Generative gaussian splatting for efficient 3d content creation," in *ICLR*, 2024.

[65] A. Chen *et al.*, "Lara: Efficient large-baseline radiance fields," in *ECCV*, 2024.

[66] Q. Wang *et al.*, "Shape of motion: 4d reconstruction from a single video," 2024.

[67] J. Ren *et al.*, "L4gm: Large 4d gaussian reconstruction model," in *Advances in Neural Information Processing Systems*, December 2024.

[68] R. Wu *et al.*, "Cat4d: Create anything in 4d with multi-view video diffusion models," *CVPR*, 2025.