Ukrainian Catholic University

Faculty of Applied Sciences

Computer Science Program

Text-to-3D using 2D Diffusion and Gaussian Splatting

MMML Final Project Report

Authors:
Yuliia Moliashcha
Taras Lysun
Andrii Ahitoliev

7 May 2025

Abstract

In this work, we provide an overview of optimization-based text-to-3D generation methods using Gaussian Splatting. We explore how pre-trained 2D diffusion models can guide the creation of 3D content without requiring extensive paired text-3D datasets. We conduct a systematic comparison of two representative approaches in this domain: GSGen and DreamGaussian. We examine their architectural differences, particularly in initialization strategies and optimization techniques. Through theoretical examination and experimental validation, we identify their respective advantages and suggest directions for future research. Our code is publicly available on GitHub [17].

1 Introduction

1.1 Motivation

Text-to-3D synthesis remains a challenging problem at the intersection of computer vision and computer graphics. While the emergence of large-scale open datasets for training image-text models [13], coupled with recent advancements in generative models, has enabled high-fidelity and diverse image synthesis, extending them to 3D generation introduces challenges due to the inherent complexity of 3D data and real-world scenes; this complexity makes dataset creation particularly difficult, and as a result it is hard to achieve realistic and varied 3D objects with generative models as they require substantial amounts of high-quality data for good generalization [19].

To address these limitations, methods that leverage existing pre-trained 2D diffusion models to guide the optimization of 3D representations have been proposed. DreamFusion [11] pioneered this approach by proposing Score Distillation Sampling (SDS), a technique that optimizes Neural Radiance Fields (NeRF) such that their renderings align with the distribution of images learned by text-to-image diffusion models. This approach transfers knowledge from 2D generative models to 3D representations, bypassing the need for extensive paired text-3D datasets. However, NeRF-based methods face computational challenges in both rendering speed and optimization convergence. Recent research has explored Gaussian Splatting as an alternative 3D representation, modeling scenes as collections of explicit 3D Gaussian primitives rather than implicit neural networks. This approach offers improvements in rendering efficiency while maintaining visual quality, making it particularly promising for text-to-3D generation.

1.2 Contributions

In this work, we compare two methods, GSGen and DreamGaussian, that implement optimization-based text-to-3D generation using Gaussian Splatting. Through analysis of their theoretical foundations, implementation strategies, and performance across evaluation metrics, we provide systematic evaluation of these approaches and identify directions for future research in text-guided 3D content generation.

1.3 Structure of the Work

This work is organized as follows: In Section 2, we provide the necessary background on diffusion models, Neural Radiance Fields (NeRF), and Gaussian Splatting techniques

that form the foundation of text-to-3D generation. In this Section we also present a comprehensive overview of existing text-to-3D approaches, including feedforward generation methods, optimization-based methods and view reconstruction techniques. In Section 3, we detail our experimental framework, evaluation metrics, and present comparisons of two proposed methods. Finally, Section 4 summarizes our findings, discusses limitations of current approaches, and suggests directions for future research.

2 Method Discussion

2.1 Diffusion Models

Early generative models, such as Variational Autoencoders (VAEs), flow-based models, and Generative Adversarial Networks (GANs) [3], laid the groundwork for image synthesis. Each of these approaches brought advances to the field while also exposing inherent limitations: VAEs produced blurry outputs, flow-based models faced architectural constraints, and GANs suffered from instability and mode collapse.

At one point diffusion models outperformed GANs [2]. Subsequently, several foundational diffusion works have been proposed such as [4, 14, 15].

Diffusion models, essentially inspired by non-equilibrium statistical physics, define a Markov-chain of diffusion steps, which involve the process of systematically and slowly destroying data in data distribution, by adding noise which is defined by iterative forward diffusion process and learning reverse diffusion process that restores structure in data [14].

In the forward diffusion process, we begin with an image sample x_0 drawn from real data distribution and iteratively add small amount of Gaussian noise over T time steps (e.g. 1000) according to a predefined variance schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$.

The forward diffusion process is defined by the following Markov chain:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

and the joint distribution over all steps is given by:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

The final sample x_T approaches a standard normal distribution, i.e.,

$$q(x_T|x_0) \approx N(0, I)$$

by the Law of Large Numbers.

Adding noise to an image is not a reversible operation; however, we can approximate the reverse conditional distribution. If parameter β is small, the effect of noise addition can be approximated by a normal distribution with unknown parameters (mean and standard deviation). The reverse diffusion process, therefore, learns to estimate these parameters with respect to current sample x_t and the timestep t.

We can write this process as a Markov chain:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

and the joint probability of a sequence of samples can be written as a product of conditional and marginal probabilities:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

A drawback is that the number of steps needed is large, and we cannot parallelize this process as it is sequential. Also we need to estimate parameters for each pixel.

Standard diffusion models run directly in pixel space, at each step we approximate noise of a size n x n x 3 and subtract it, which is booth memory- and compute-intensive. Thus Diffusion models require optimization – Latent Diffusion Models [12], where instead of running a diffusion process in pixel space we run it in the latent space.

In this work we focus on latent diffusion models, specifically we use an open-source guided latent diffusion model Stable Diffusion [18].

2.2 Neural Radiance Fields & Gaussian Splatting

Radiance Fields The field of view synthesis has been significantly advanced by Radiance Field methods, which have revolutionized the visualization of scenes captured through multiple photographs or videos. Radiance Fields function by creating a volumetric representation of a scene. Their goal is to generate highly realistic 3D scenes from a collection of 2D images, enabling the viewing of an object or scene from virtually any angle. Many radiance field techniques initiate their process by analyzing standard 2D images using a Structure from Motion (SfM) algorithm [10]. SfM aligns these images and produces sparse 3D point clouds that guide the training of the radiance field model.

Among the different radiance field techniques, Neural Radiance Fields (NeRF) [8] stand out as a creative method. NeRF uses a deep neural network to represent a volumetric scene function. This network takes a 5D coordinate as input (3D spatial location and a 2D viewing direction) and returns the volume density and emitted radiance at that point in the scene. To train a NeRF model, it uses backpropagation to learn how the input coordinates relate to the output color and density. Once the model is trained, it can create images of the scene from new angles. However, NeRF has a major downside: it needs a lot of computing power and time to train and render. Real-time rendering with NeRF usually requires powerful hardware. Also, NeRF works best with static scenes. If the scene changes, you usually have to retrain the model completely to keep the 3D representation accurate.

Addressing the limitations of NeRF, Gaussian Splatting was developed as a faster and more efficient alternative for achieving real-time rendering [6]. Unlike NeRF, which uses an implicit representation through a neural network, Gaussian splatting employs an explicit representation based on rasterization of Gaussian primitives. This approach allows Gaussian splatting to maintain the high visual fidelity and view-dependent effects characteristic of Radiance Fields, while also achieving real-time rendering performance, even on mobile devices. In many instances, Gaussian splatting offers comparable or even superior rendering quality to NeRF, with the added benefits of faster training and inference times. Its efficiency makes it particularly suitable for interactive applications such as live simulations and virtual or augmented reality experiences.

Gaussian primitives In Gaussian Splatting, a 3D scene is represented as a dense collection of individual 3D Gaussian functions. This representation involves millions of these Gaussian particles populating the 3D space. Each Gaussian effectively models a volumetric "splat" within the scene. This approach draws an analogy to traditional triangle rasterization in computer graphics, but instead of using triangles as the fundamental rendering primitives, it employs these Gaussians. Figure 1 illustrates the overall pipeline of 3D Gaussian Splatting, from input images to the optimized 3D Gaussian representation [6]

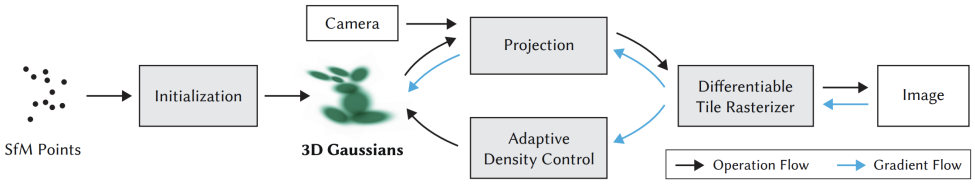

Figure 1: Overview of the 3D Gaussian Splatting pipeline, showing the process from input images to optimized 3D Gaussian representation [6]

Each of these 3D Gaussian primitives is defined by a set of parameters that describe its properties and its contribution to the final rendered image:

- Mean (μ): central position of the Gaussian in 3D space, specified by its coordinates in R^3.
- Covariance Matrix (Σ): A 3x3 matrix showing how the Gaussian stretches or scales along different axes, allowing for an anisotropic (non-uniform) representation.
- Opacity (α): controls the transparency of the Gaussian, ranging from 0 (fully transparent) to 1 (fully opaque). It determines the extent to which the Gaussian obscures objects behind it, contributing to the lifelike quality of the scene.
- Spherical Harmonic (SH) Coefficients: These coefficients are used to represent the view-dependent color information of the Gaussian. They model the directional appearance (color) of the radiance field, allowing the color of a Gaussian to change based on the viewing angle. The entire radiance field of the scene is approximated by this collection of Gaussians, each associated with its own color information encoded through spherical harmonics.

The mathematical foundation of Gaussian splatting relies on representing each element of a 3D scene as a 3D Gaussian function. The general form of a multivariate Gaussian function in 3D is given by:

$$G(x) = e^{-\frac{1}{2}(x)^T \Sigma^{-1}(x)}$$

In the context of rendering, this Gaussian value is often multiplied by the opacity α of the Gaussian during the blending process to determine its contribution to the final pixel color.

The covariance matrix Σ_i is a crucial component that defines the ellipsoidal shape of the Gaussian. It can be decomposed into a scaling matrix and a rotation matrix, allowing for intuitive control over the Gaussian's form and orientation. The scaling is determined by a scaling vector s_i, which specifies the scale along the three principal axes (s_x, s_y, s_z). This vector can be represented as a diagonal matrix $S(s_i)$. The orientation of the Gaussian is defined by a rotation quaternion q_i, which is a four-dimensional number (q_r, q_i, q_j, q_k) used to efficiently represent 3D rotations. This quaternion can be converted into a 3x3 rotation matrix $R(q_i)$ that describes the Gaussian's orientation in 3D space.

The covariance matrix Σ_i is then derived from these scaling and rotation components using the formula:

$$\Sigma_i = R(q_i)S(s_i)S(s_i)^T R(q_i)^T$$

Since the scaling matrix $S(s_i)$ is a diagonal matrix, the formula often simplifies to $\Sigma_i = R(q_i)S(s_i)S(s_i)R(q_i)^T$ or equivalently, $\Sigma_i = R(q_i)S(s_i)^2R(q_i)^T$. This mathematical expression illustrates how the covariance matrix is constructed by first applying a scaling transformation to a unit sphere (which has an identity covariance matrix) and then rotating the resulting ellipsoid to the desired orientation. It is important to note that the covariance matrix is always symmetric and positive definite, because it must represent a valid ellipsoid.

The decomposition of the covariance matrix into scaling and rotation components offers an intuitive way to manipulate the shape and orientation of each Gaussian primitive. Instead of directly dealing with the nine elements of the covariance matrix, it is more straightforward to think in terms of scaling along specific axes and rotation around an axis. This parameterization is not only more physically interpretable but also enables the optimization process during training of the Gaussian Splatting model. The formula for the covariance matrix also highlights the sequence of transformations applied: the scaling is performed first, followed by the rotation.

2.3 Previous Approaches

DreamFusion [11] introduces a new approach to text-to-3D synthesis by using pre-trained 2D text-to-image diffusion models, thereby eliminating the need for extensive 3D datasets. Instead of using CLIP-based guidance (as seen in prior methods like Dream Fields), DreamFusion introduces Score Distillation Sampling (SDS), a loss function that enables optimization in parameter space instead of pixel space. SDS operates by aligning the 3D model's parameters so that its 2D renderings conform to the distributions learned by the 2D diffusion model. This is achieved by computing gradients in the parameter space of the 3D model. As a result the 3D model inherits the visual fidelity of the 3D model without the direct 3D supervision, thus enabling the generation of high-quality and visually consistent 3D objects from textual descriptions. Neural Radiance Fields (NeRF), utilized in DreamFusion, provide a way to model 3D scenes implicitly through volume rendering, enabling high-quality reconstructions from sparse views. However, NeRFs require extensive sampling along rays and iterative optimization, making inference slow and capturing fine details challenging. The principles of SDS have been further extended in subsequent works, such as DreamGaussian. DreamGaussian employs a generative 3D Gaussian Splatting model, which, when combined with mesh extraction and texture refinement techniques, significantly accelerates the 3D content creation process. Gaussian Splatting, on the other hand, instead of relying on an implicit neural network to encode scene geometry, uses a collection of Gaussians to model both the spatial structure and the appearance of the scene. This explicit formulation allows for the direct incorporation of 3D priors during the geometry optimization and texture refinement stages, often resulting in sharper, more detailed meshes and significantly faster inference times. This makes Gaussian Splatting much more efficient for applications demanding both high mesh quality and fast processing.

2.4 Text-to-3D generation

Text-to-3D generation represents a significant advance in computational graphics that leverages pre-trained text-to-image diffusion models to construct 3-dimensional assets from textual descriptions.

The challenge in text-to-3D generation lies in efficiently translating the high-level semantic information present in textual descriptions into the precise geometric and visual details that define a 3D object. Traditional 3D modeling demands manual intervention, often requiring specialized skills and software for tasks such as sculpting, texturing, and rigging. Text-to-3D methodologies aim to automate this process by learning a mapping from the textual domain to the 3D domain. Based on recent studies of the field, the primary methods can be generally classified into feedforward generation, optimization-based generation, and view reconstruction approaches.

2.4.1 View Reconstruction

View reconstruction methods [7] for text-to-3D generation operate by first generating multiple 2D views of the desired object from a text prompt and subsequently reconstructing a 3D model from the initial stage involves employing a powerful text-to-image model, often based on diffusion models, to generate several images of the object described by the text prompt, potentially from different virtual camera viewpoints. These text-to-image models are trained on vast datasets of images and their corresponding textual descriptions, learning a complex mapping from text embeddings to image pixel space. Mathematically, a diffusion model for image generation learns to reverse a process that gradually adds noise to an image x_0 over T steps, resulting in a completely noisy image $x_T N(0, I)$. The forward process can be defined as:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}),$$

where β_t is a variance schedule. The reverse process, which generates an image from noise, is also a Markov chain:

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)),$$

where the mean and variance are learned by a neural network parameterized by . In text-conditioned generation, the text prompt is typically encoded into embeddings which are then used to condition the denoising process at each step. Once a set of 2D views $I_1, I_2, ..., I_n$ is generated, the next stage is to reconstruct a 3D model. This can be achieved using traditional Multi-View Stereo (MVS) techniques or more recent neural reconstruction methods.

Traditional Multi-View Stereo (MVS): MVS aims to find the corresponding points in multiple images and use triangulation to estimate their 3D positions. Given two camera projection matrices P_1 and P_2 and a pair of corresponding image points x_1 and x_2, the 3D point X projected to these points satisfies $P_1X = x_1$ and $P_2X = x_2$. Solving this system of equations (often overdetermined with multiple views) yields the 3D coordinates of the point.

The process involves:

- Feature Matching: Identifying corresponding features (e.g. corners, edges) in the generated 2D views. This can be done using feature detectors like SIFT or SURF, followed by matching based on feature descriptors.

- Camera Pose Estimation: Determining the position and orientation of the virtual cameras used to generate the 2D views. If the camera parameters are known from the text-to-image generation process, this step is simplified. Otherwise, techniques like Structure from Motion (SfM) might be used to jointly estimate camera poses and 3D structure.

- Depth Map Estimation: For each reference view, estimate the depth of each pixel by finding its correspondence in other views. This often involves defining a cost function based on photometric consistency (similarity of pixel colors across views after accounting for camera geometry) and regularizing the depth map to be smooth.
- Point Cloud Fusion: Combining the depth maps from multiple views into a dense 3D point cloud. This involves transforming the depth information into 3D coordinates based on the camera parameters.

2.4.2 Feedforward generation

This category encompasses methods that aim to directly produce a 3D representation from a text prompt using a neural network that has been trained to learn this mapping. These techniques typically involve training a generative model on a substantial dataset of 3D assets paired with corresponding textual descriptions. Examples of effective feedforward methods include Point-E, which generates point clouds; Shap-E, which produces parameters for implicit functions; and MeshDiffusion, which directly synthesizes 3D meshes [5].

Shap-E: Generating Conditional 3D Implicit Functions This method directly generates the parameters of implicit functions conditioned on a text prompt. An implicit function defines a 3D shape as the set of all points $x = (x, y, z)$ in R^3 that satisfy the equation $F(x; \theta) = 0$, where θ represents the parameters of the function.

The training of Shap-E occurs in two stages. First, an encoder network is trained to deterministically map existing 3D assets (represented as textured meshes or neural radiance fields) into the parameter space of the implicit function. This encoder learns to compress the 3D information into a latent representation, which is then linearly projected to obtain the weights of the MLP representing the implicit function. Second, a conditional diffusion model is trained on the outputs of this encoder, i.e., the parameters , conditioned on the corresponding text descriptions.

The diffusion process in Shap-E involves gradually adding Gaussian noise to the latent parameters over a series of time steps. The model then learns to reverse this process, starting from a random noise vector, to generate new sets of parameters that correspond to a 3D shape described by the input text prompt.

Point-E: Generating 3D Point Clouds from Complex Prompts Another feedforward approach is Point-E , which focuses on directly generating 3D point clouds from text prompts. A point cloud is a set of discrete points in R^3 that collectively represent the surface of an object. The architecture of Point-E likely involves a generative model, potentially a diffusion model or a VAE, that is trained to predict the coordinates of these points conditioned on the input text.

The training process for such a model would involve a loss function that quantifies the similarity between the generated point clouds and the ground truth point clouds from the training data. Common metrics for comparing point clouds include Chamfer distance and Earth Mover's distance, which measure the average distance between the points in two sets.

MeshDiffusion: Score-based Generative 3D Mesh Modeling MeshDiffusion represents a feedforward method that directly generates 3D meshes from text prompts. A

3D mesh is an explicit representation of an object's surface, defined by a set of vertices in 3D space and a collection of faces (typically triangles or polygons) that connect these vertices. As a score-based generative model, MeshDiffusion employs a diffusion process operating directly in the space of 3D meshes.

2.4.3 Optimization-based Text-to-3D Generation

DreamFusion Optimization-based text-to-3D generation methods, exemplified by DreamFusion, leverage the power of pretrained 2D text-to-image diffusion models to guide the creation of 3D content. DreamFusion specifically utilizes a pre-trained 2D diffusion model, such as Imagen, to optimize a 3D scene represented by a Neural Radiance Field (NeRF) based on a given text prompt, as illustrated in Figure 2.

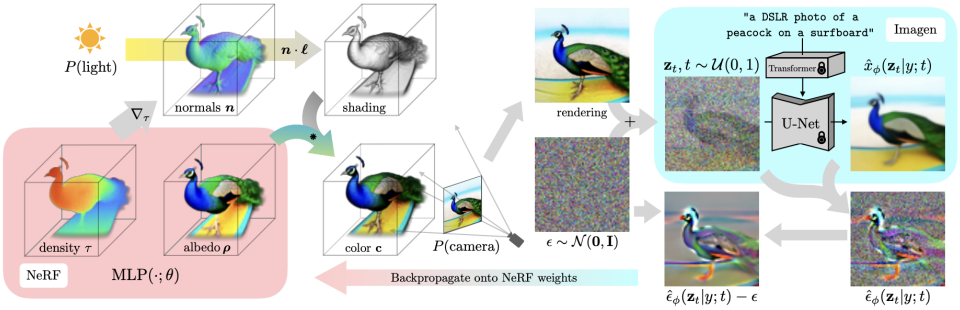

Figure 2: Proposed DreamFusion framework [11]

NeRFs: At the core of DreamFusion is the representation of a 3D scene using a NeRF. A NeRF models a 3D scene as a continuous volumetric function that can be queried at any 3D coordinate $x = (x, y, z)$ and viewing direction $d = (\theta, \phi)$ to produce the color c and volume density at that point. Mathematically, this can be expressed as $NeRF : (x, d) \rightarrow (c, \sigma)$. The volume density σ determines the probability that a ray will end at that point, and the color c represents the radiance emitted. The color of each ray is then computed by integrating the color and density values along the ray using numerical quadrature techniques.

Score Distillation Sampling (SDS): DreamFusion introduces a novel loss function called SDS to optimize the parameters Θ of the NeRF based on the guidance of the 2D diffusion model. The fundamental idea behind SDS is to leverage the score function learned by the pre-trained 2D diffusion model. The score function $s_\theta(x, t)$ estimates gradient of the density of the log data of the image distribution at a given noise level t. Given a text prompt, the SDS loss aims to optimize the NeRF parameters such that the 2D renderings $I(\Theta)$ of the NeRF from various viewpoints align with the distribution of images conditioned on the text, as learned by the diffusion model. The SDS loss can be formulated as:

$$\nabla_\Theta \mathcal{L}_{\text{SDS}} = E_{t,p,\epsilon}\left[w(t)(\epsilon_\phi(\mathbf{x}; t, e) - \epsilon)\frac{\partial \mathbf{x}}{\partial \Theta}\right]$$

where p is a random camera pose, ϵ is a sample of standard Gaussian noise $N(0, I)$, t is a randomly sampled timestamp, $w(t)$ is a weighting function, $\epsilon_\phi(\mathbf{x}; t, e)$ is the noise predicted by the diffusion model at time t for image $\mathbf{x}$ conditioned on text embedding e.

By minimizing the difference between this score and noise, the optimization process encourages the NeRF renderings to lie in the data manifold learned by the 2D diffusion model.

The optimization process in DreamFusion involves iteratively performing the following steps: randomly sample a camera pose p; render a 2D image $I(\Theta)$ of the NeRF from this viewpoint; add Gaussian noise $\sigma_t \epsilon$ to the rendered image at a random noise level t; compute the SDS loss using the score function of the pre-trained 2D diffusion model; and update the parameters Θ of the NeRF using gradient descent to minimize SDS loss. This process is repeated until the NeRF converges to a 3D representation that, when rendered from various angles, produces images that are consistent with the input text prompt according to the 2D diffusion model.

DreamGaussian & GSGen Both DreamGaussian and GSGen employ Gaussian primitives for 3D representation but differ in their initialization strategies and optimization priors.

DreamGaussian initializes from a sphere of particles, distributing Gaussian primitives uniformly across a spherical volume with unit scaling and no rotation (Figure 3). This approach relies on the optimization process itself to develop correct geometry without external priors [16]. In contrast, GSGen leverages Point-E [9], a conditional 3D point cloud diffusion model, to generate a structured initial distribution of points that better approximates the target geometry (Figure 4). This initialization provides a stronger geometric prior, helping to mitigate common issues like the Janus problem where models develop inconsistent geometry or multiple faces [1].

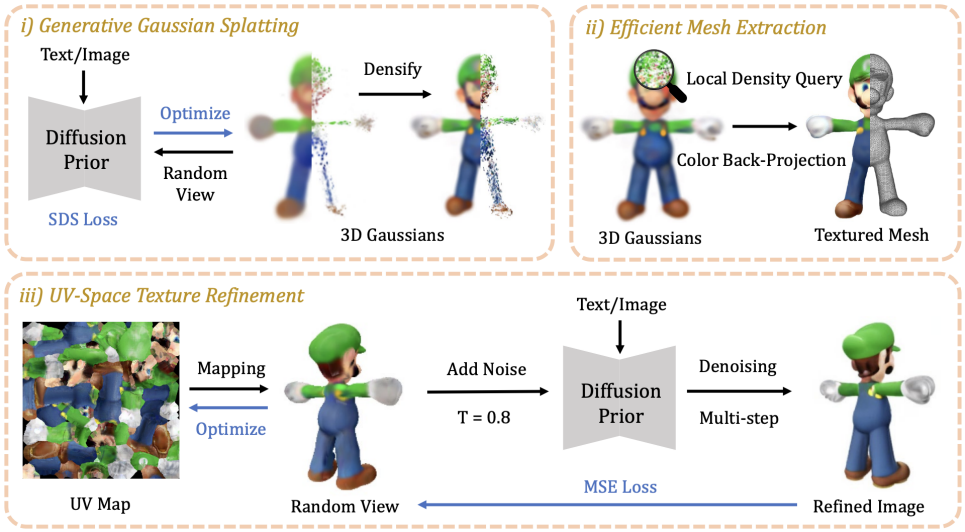

Figure 3: Proposed DreamGaussian framework, utilizing a sphere of randomly positioned particles with unit scaling and no rotation prior and two-stage optimization process (geometry optimization and appearance refinement). The pipeline also includes Mesh Extraction and UV-Space Texture Refinement to generate textured 3D mesh assets [16]

GSGen's two-stage optimization process consists of: geometry optimization, where Gaussian parameters are optimized under the joint guidance of a 3D point cloud diffusion prior and 2D image diffusion prior through SDS; and appearance refinement, where details are enriched with compactness-based densification [1].

DreamGaussian similarly employs a two-stage approach, but with different emphases: it first initializes and optimizes 3D Gaussians using SDS loss without explicit 3D priors [16]. Rather than continuing refinement within the Gaussian representation, DreamGaussian extracts a textured mesh from the optimized Gaussians through local density

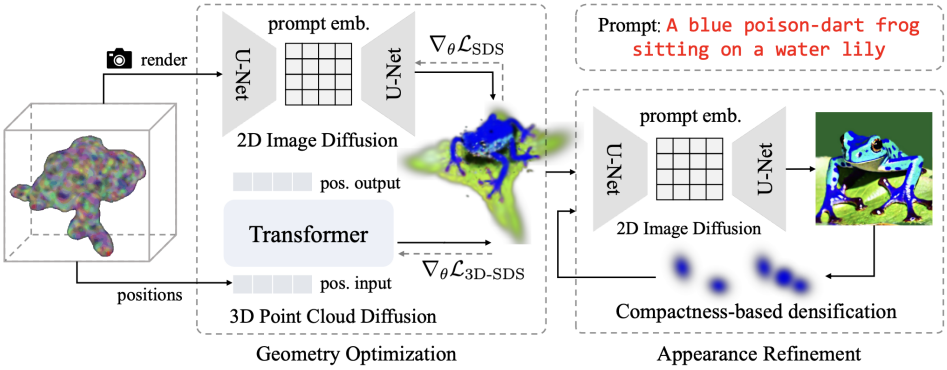

Figure 4: Proposed GSGEN framework, utilizing Point-E prior and two-stage optimization process (geometry optimization and appearance refinement) [1]

querying and refines the texture in UV space using differentiable rendering and multi-step MSE loss to avoid artifacts.

3 Experiments & Results

In our experimental evaluation, we focus on two representative methods, GSGen and DreamGaussian, that represent optimization-based approaches using Gaussian Splatting. These methods were selected for comparative analysis as they utilize similar underlying principles but implement distinct strategies for text-to-3D generation.

3.1 Experimental Setup

We used a guidance model based on the publicly available diffusion model, Stable Diffusion [12, 18], using the checkpoint `runwayml/stable-diffusion-v1-5`.

3.2 Qualitative & Quantitative Comparisons

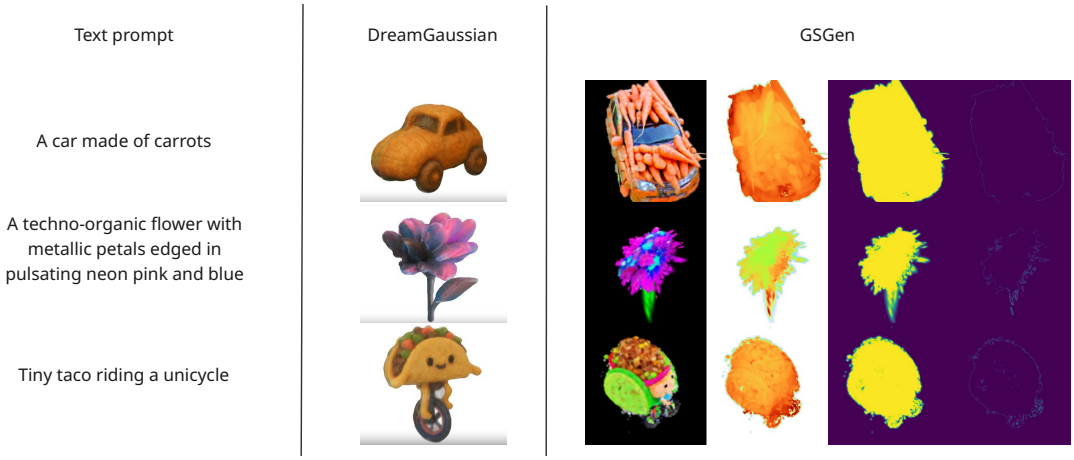

Figure 5: Qualitative results

We render images from 60 viewpoints with a uniform azimuth angle and average results and calculate the metrics.

Metric	DreamGaussian	GSGen
PSNR	17.39	23.50
SSIM	0.827	0.890
LPIPS	0.232	0.140
CLIP-Similarity (R-Precision)	0.833	0.870
Generation Time	∼1min	∼40min

Table 1: Quantitative comparison between DreamGaussian and GSGen

.

4 Conclusions

In this work, we conducted a systematic comparison of optimization-based text-to-3D generation methods using Gaussian Splatting. Our analysis revealed that GSGen's Point-E initialization yields superior visual quality, while DreamGaussian offers computational efficiency advantages. Both approaches leverage pre-trained 2D diffusion models without requiring paired text-3D datasets, demonstrating improvements over NeRF-based methods in rendering efficiency. These findings suggest that initialization strategy plays an important role in the quality-performance trade-off for text-guided 3D synthesis. Future work should explore advanced diffusion models, Variational Score Distillation, and optimization acceleration techniques to further advance text-to-3D generation.

References

[1] Zilong Chen, Feng Wang, Yikai Wang, and Huaping Liu. Text-to-3d using gaussian splatting, 2023.

[2] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[5] Chenhan Jiang. A survey on text-to-3d contents generation in the wild, 2024.

[6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering, 2023.

[7] Jian Liu, Xiaoshui Huang, Tianyu Huang, Lu Chen, Yuenan Hou, Shixiang Tang, Ziwei Liu, Wanli Ouyang, Wangmeng Zuo, Junjun Jiang, and Xianming Liu. A comprehensive survey on 3d content generation, 2024.

[8] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[9] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts, 2022.

[10] Onur Ozyesil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion, 2017.

[11] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion, 2022.

[12] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[13] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models, 2022.

[14] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[15] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution, 2019.

[16] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation, 2023.

[17] taraslysun. taraslysun/splatfusion3d. `https://github.com/taraslysun/SplatFusion3D`, 2025.

[18] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`, 2022.

[19] Zhen Wang, Dongyuan Li, Yaozu Wu, Tianyu He, Jiang Bian, and Renhe Jiang. Diffusion models in 3d vision: A survey, 2024.