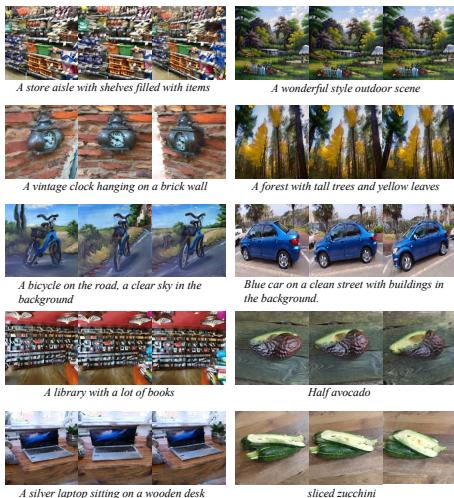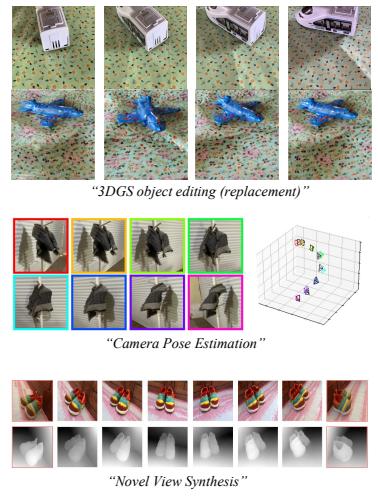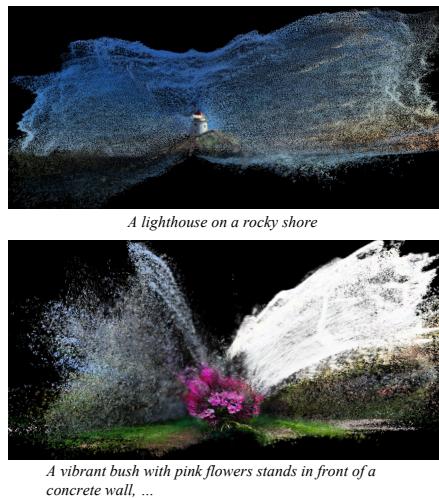# SplatFlow: Multi-View Rectified Flow Model for 3D Gaussian Splatting Synthesis

Hyojun Go[1*]   Byeongjun Park[2*]   Jiho Jang[1]   Jin-Young Kim[1]

Soonwoo Kwon[1]   Changick Kim[2†]

[1]Twelve Labs   [2] KAIST

A store aisle with shelves filled with items

A wonderful style outdoor scene

A vintage clock hanging on a brick wall

A forest with tall trees and yellow leaves

A bicycle on the road, a clear sky in the background

Blue car on a clean street with buildings in the background.

A library with a lot of books

Half avocado

A silver laptop sitting on a wooden desk

sliced zucchini

A lighthouse on a rocky shore

A vibrant bush with pink flowers stands in front of a concrete wall, ...

"3DGS object editing (replacement)"

"Camera Pose Estimation"

"Novel View Synthesis"

(a) 3D Gaussian Splatting Synthesis

(b) Training-Free Applications

Figure 1. **SplatFlow for 3D Gaussian Splatting synthesis and its training-free applications.** (a) Examples of direct 3D Gaussian Splatting (3DGS) generation only from text prompts, (b) Training-free applications, including 3DGS object editing, camera pose estimation, and novel view synthesis. SplatFlow seamlessly integrates these capabilities, showcasing its versatility in generating and editing complex 3D content.

Abstract

Text-based generation and editing of 3D scenes hold significant potential for streamlining content creation through intuitive user interactions. While recent advances leverage 3D Gaussian Splatting (3DGS) for high-fidelity and real-time rendering, existing methods are often specialized and task-focused, lacking a unified framework for both generation and editing. In this paper, we introduce Splat-Flow, a comprehensive framework that addresses this gap by enabling direct 3DGS generation and editing. Splat-Flow comprises two main components: a multi-view rectified flow (RF) model and a Gaussian Splatting Decoder (GSDecoder). The multi-view RF model operates in latent space, generating multi-view images, depths, and camera poses simultaneously, conditioned on text prompts—thus addressing challenges like diverse scene scales and complex camera trajectories in real-world settings. Then, the GSDecoder efficiently translates these latent outputs into 3DGS representations through a feed-forward 3DGS method. Leveraging training-free inversion and inpainting techniques, SplatFlow enables seamless 3DGS editing and supports a broad range of 3D tasks—including object editing, novel view synthesis, and camera pose estimation—within a unified framework without requiring additional complex pipelines. We validate SplatFlow's capabilities on the MVImgNet and DL3DV-7K datasets, demonstrating its versatility and effectiveness in various 3D generation, editing, and inpainting-based tasks. Our project page is available at https://gohyojun15.github.io/SplatFlow/.

1. Introduction

The demand for realistic 3D scene generation and editing from text has surged in applications like VR/AR, gaming, and robotics. Early works primarily utilized NeRF [64, 68] for scene generation [119] and editing [3, 34, 35, 39, 76, 115], but NeRF's volumetric rendering is computationally

---

*Equal contribution

†Corresponding author

1

intensive, resulting in slower rendering speeds. Recently, 3D Gaussian Splatting (3DGS) [38] has emerged as a promising alternative, offering real-time and high-fidelity rendering implementations. Building on this, recent advances in 3D generation [11, 40, 71, 97, 104, 105, 109, 111, 118, 121, 127] and editing [11, 40, 71, 104, 105, 109, 127] increasingly leverage 3DGS to achieve both speed and quality.

Despite these advancements, current methods for generating and editing 3DGS are specialized and task-focused, lacking a unified framework. For 3DGS generation, several works [13, 50, 57, 98, 100, 113] leverage 2D diffusion models [31, 85, 92, 95] with Score Distillation Sampling (SDS) [80], which requires time-intensive per-scene optimization. To address this, recent studies have shifted towards direct 3DGS generation, combining diffusion models with reconstruction models [97, 111, 121] or utilizing 3D generative models [118]. However, most of these methods are restricted to object-level 3D generation using synthetic datasets [18, 19] with controlled, bounded environments. In contrast, real-world scenes, varying in scene scales and camera trajectories, present distinctive challenges.

In 3DGS editing, several works [11, 40, 71, 104, 105, 109, 127] adopt 2D diffusion models [31, 85, 92, 95] to guide the editing process. The main challenge is lifting 2D editing guidance to 3D while maintaining 3D-consistent modifications. One approach focuses on SDS techniques to update 3DGS [71, 110, 127], but these methods require additional stages like texture adjustments [71, 110] and fine-grained refinements [127]. Another approach aims to edit multi-view images with 3D consistency [40, 105, 109], introducing attention-based modules [105, 109] and autoregressive editing [40], adding complexity to the editing pipeline.

In this paper, we bridge the gap between 3DGS generation and editing by designing a direct 3DGS generative model. Inspired by the success of 2D diffusion models that enable training-free editing through inversion [16, 62, 67], we argue that a direct 3DGS model can similarly enable training-free editing. This approach integrates editing without requiring complex pipelines to lift 2D priors, making generation and editing adaptable within a unified framework.

We introduce SplatFlow, which consists of two main components: (1) a multi-view rectified flow (RF) model and (2) a Gaussian Splatting Decoder (GSDecoder). Similar to Latent Diffusion Models [85], our multi-view RF model operates in the latent space and is trained to simultaneously generate multi-view images, depths, and camera poses conditioned on text prompts. This joint modeling enables SplatFlow to effectively handle the challenges of real-world scene generation, such as various scene scales and camera trajectories, unlike synthetic object datasets [18, 19]. The GSDecoder translates the latent outputs into 3DGS, based on efficient feed-forward methods [6, 12]. To ensure compatibility with other generative models, particularly Stable Diffusion 3 [22],

we incorporate a fixed pre-trained encoder, allowing flexible cross-model usage during sampling.

Finally, SplatFlow utilizes training-free inversion [16, 62, 67] and inpainting techniques [14, 61, 86, 87, 93] for 3DGS editing and various 3D tasks. For example, it enables object editing [9, 71], novel view synthesis [74, 84, 116], and camera pose estimation [120] by jointly modeling images, depths, and poses to infer missing elements, highlighting its versatility. We validated its capabilities using the MVImgNet [117] and DL3DV-7K [54] datasets, assessing its performance in 3D generation, editing, and inpainting-based tasks.

2. Related Works

2.1. Diffusion Models and Rectified Flows

Diffusion Models Diffusion models [31, 92, 95] generate data through iterative denoising and have become standard in generative modeling across images [20, 24, 44, 73, 75], videos [32], and text [47]. Trained on large-scale text-to-image datasets, they capture complex semantic relationships, producing high-quality images [8, 69, 79, 85, 88]. An advantage is their adaptability to various tasks in a plug-and-play manner [23, 25, 94], especially inpainting [61, 86] and editing [27, 41]. In painting can be framed as a linear inverse problem [17], for which effective posterior sampling methods have been proposed [14, 61, 86, 87, 93]. For editing, inversion methods [16, 62, 67] produce structured noise by inverting images, allowing new prompts to be effectively processed. Building on these training-free techniques, we extend them to our 3DGS model, adapting RePaint [61] and SDEdit [62] within rectified flow models.

Rectified Flow Models Rectified Flow (RF) models [2, 55, 56] represent an alternative approach to traditional diffusion models by utilizing a straight-line path from data to noise instead of the typical curved forward process, aiming to simplify sampling and reduce computational costs. This linear path theoretically enables faster sampling by minimizing error accumulation across steps, and recent advances, such as Stable Diffusion 3 (SD3) [22], have successfully scaled RF in text-to-image generation, demonstrating performance that surpasses traditional diffusion models. Our SplatFlow framework builds on the capabilities of SD3, integrating RF to achieve efficient 3DGS generation.

2.2. 3D Generative Models

Neural Scene Representation Various neural scene representations have been proposed for 3D generation tasks. Early works use explicit representations, enabling synthesis of 3D point clouds [70] and shapes [99, 125], but struggle with realistic scene generation due to their rendering mechanisms. To address this, NeRF [64] is employed for a realistic 3D generation with per-scene optimization [80], but its inefficient rendering is unsuitable for a fast generation.

Recently, 3DGS [38] enables real-time rendering, accelerating optimization-based 3D generation [13, 53, 98]. Building on this progress, we leverage 3DGS's efficient rendering for fast training of a generative model on large datasets.

Lifting Multi-View Generation Advances in image generation models [8, 79] and large-scale 3D object datasets [18, 19] make multi-view generation compelling compared to optimization-based generation [98]. The key idea is to learn 3D priors as conditions [58, 59, 108] or attentions [90, 101] from 3D datasets while leveraging text-to-image diffusion models [85] trained on billions of text-image pairs [89]. This enables the synthesis of multi-view consistent images from an image or text prompt, which is then fed into reconstruction models [7, 97] to directly generate neural scene representations like 3DGS. One critical success factor in these methods is that the training dataset provides rendered images from consistent, canonical viewpoints, allowing instant access to desired views regardless of the 3D object. In this work, we focus on more challenging unbounded 3D real-world scene generation, where each scene varies in scale and camera trajectory. To handle diverse trajectories, our model learns a joint distribution of camera poses and 3D scenes.

Text-to-3D Scene Generation Generating 3D scenes from text prompts is challenging due to the unbounded scale of scenes, complicating scale determination. One straightforward approach is to create a scene covered by a single image [91, 126] generated from text-to-image diffusion models [8, 79]. Another line of work [15, 33] generates wider scenes using user-provided camera trajectories, but these methods are time-consuming and produce subpar quality due to reliance on 2D generative models. Recently, Director3D [49], the work most similar to ours, generates camera poses from text and then creates multi-view images decoded into 3DGS. In contrast, SplatFlow jointly learns the distribution of camera poses and multi-view images from text, enhancing generation quality and enabling flexible 3D scene editing in a unified framework.

2.3. 3D Editing

In recent years, NeRF-based methods [3, 34, 35, 39, 72, 76, 83, 114, 115] have gained popularity for 3D editing, surpassing point clouds and meshes [102, 122, 124]. However, due to the inefficient rendering, these methods are time-consuming and lack precise control. Conversely, incorporating 3DGS into 3D editing [11, 104] improves effectiveness, speed, and control.

For guiding 3D editing, several methods have leveraged 2D diffusion models [31, 85, 92, 95] for structured 3D editing, but it often requires additional refinement stages. For instance, GSEdit [71] and Xiao *et al.* [110] apply texture enhancement, while TIP-Editor [127] adds fine-grained adjustments to reduce SDS artifacts. Another line of research in 3D editing emphasizes achieving multi-view consistency

by sequentially updating each view. For example, GaussCtrl [109] and 3DEdit [105] utilize attention-based modules to propagate edits made from one perspective across multiple views, ensuring that changes are consistently reflected throughout the 3D scene. Similarly, 3D-Ego [40] adopts an autoregressive editing framework, updating each view to enhance multi-view consistency.

Unlike prior methods that require complex procedures to lift 2D priors for 3D consistency, SplatFlow directly models multi-view consistency within the latent space, efficiently decoding edits through our Gaussian Splatting Decoder.

3. Preliminary: Rectified Flows

Rectified Flows (RFs) [2, 55, 56] consider generative models that construct a noise distribution q_0 and a time-varying vector field $v_t(\boldsymbol{x}_t)$ to sample data x_0 from a data distribution p_0 via an ordinary differential equation (ODE) as:

$$\mathrm{d}X_t = v(X_t)\,\mathrm{d}t, \quad X_0 \sim q_0, \quad t \in [0, 1], \qquad (1)$$

where q_0 typically follows a Gaussian distribution $\mathcal{N}(0, I)$. RF employs a neural network with parameters θ for modeling the vector field as $v_t(X_t) = -u_\theta(X_t, 1 - t)$.

To train a neural network as the vector field, RFs utilize paired samples from p_0 and q_0 (here simplified as p_1), along a straight line path $Y_t = tY_1 + (1 - t)Y_0$. This setup produces the marginal distribution of Y_t as $p_t(\boldsymbol{y}_t) = \int p_t(\boldsymbol{y}_t|\boldsymbol{y}_1)p_1(\boldsymbol{y}_1)\mathrm{d}\boldsymbol{y}_1$. Starting from an initial $Y_0 = \boldsymbol{y}_0$ and moving to a final state $Y_1 = \boldsymbol{y}_1$, a straight line path induces an ODE expressed as $\mathrm{d}Y_t = u_t(Y_t|Y_1)\mathrm{d}t$ where the vector field $u_t(Y_t|Y_1)$ represents the conditional change from $\boldsymbol{y}_0$ to $\boldsymbol{y}_1$, given by $u_t(Y_t|Y_1) = \boldsymbol{y}_1 - \boldsymbol{y}_0$. The marginal vector field $u_t(\boldsymbol{y}_t)$ is computed by averaging over the conditional vector field as [55]:

$$u_t(\boldsymbol{y}_t) = \int u_t(\boldsymbol{y}_t|\boldsymbol{y}_1)\frac{p_t(\boldsymbol{y}_t|\boldsymbol{y}_1)}{p_t(\boldsymbol{y}_t)}p_1(\boldsymbol{y}_1)\mathrm{d}\boldsymbol{y}_1. \qquad (2)$$

To approximate $u_t(\boldsymbol{y}_t)$, the neural network is trained with a flow-matching object, defined by:

$$\mathcal{L}_{\mathrm{FM}} := \mathbb{E}_{t, Y_t}\left[||u_t(Y_t) - u_\theta(Y_t, t)||_2^2\right]. \qquad (3)$$

Since directly computing this objective can be computationally challenging, an alternative approach, conditional flow matching, is used to simplify training:

$$\mathcal{L}_{\mathrm{CFM}} := \mathbb{E}_{t, Y_t, Y_1 \sim p_1}\left[||u_t(Y_t|Y_1) - u_\theta(Y_t, t)||_2^2\right]. \qquad (4)$$

The gradients of L_{CFM} and L_{FM} are theoretically identical [55], making them equivalent in training. However, L_{CFM} offers better computational efficiency and is therefore preferred. Ultimately, the learned vector field for the ODE in Eq. 1 is given by $v_t(X_t) = -u_\theta(X_t, 1 - t)$, allowing RFs to generate samples that follow the data distribution using the learned vector field.
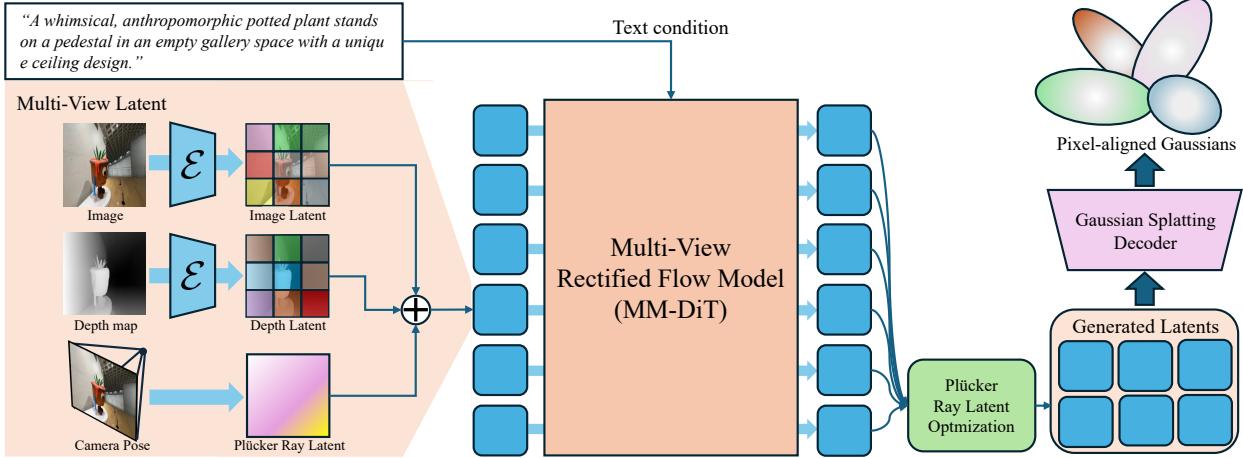
Figure 2. Overview of SplatFlow. SplatFlow consists of two main components: a multi-view Rectified Flow (RF) model and a Gaussian Splat Decoder (GSDecoder). Conditioned on text prompts, the RF model generates multi-view latents—including image, depth, and Plücker ray coordinates. After an optimization process to estimate camera poses, the GSDecoder decodes these latents into pixel-aligned 3DGS.

4. Methodology

4.1. Overview of SplatFlow

We introduce *SplatFlow*, a 3DGS generation model designed to perform various 3D editing and 3D-related tasks in a training-free manner, despite being trained solely for generation. As shown in Fig. 2, our model consists of two main components: 1) a multi-view Rectified Flow (RF) model and 2) a Gaussian Splatting Decoder (GSDecoder). First, the multi-view RF model generates multi-view camera poses, depths, and images conditioned on text prompts. Then, the GSDecoder translates these latent representations into pixel-aligned 3DGS, constructing a real-world scene structure.

Also, both the multi-view RF model and GSDecoder operate within a latent space, adopting a Latent Diffusion Model (LDM) [85]. By freezing and sharing the encoder of Stable Diffusion 3 (SD3) [22], our design enables the RF model to share the latent space with SD3, allowing us to leverage its knowledge for enhanced generation and editing capabilities.

Finally, our model leverages a training-free approach [61, 62], enabling a broad range of editing applications, such as object replacement, as well as 3D-related tasks including camera pose estimation and novel view synthesis.

In the following sections, we present each component in detail. Section 4.2 introduces the GSDecoder, explaining details in constructing pixel-aligned 3DGS. Section 4.3 describes the multi-view RF model, detailing how it jointly generates camera poses, depths, and images. Finally, Section 4.4 outlines the sampling process and demonstrates how our model applies training-free techniques.

4.2. Gaussian Splatting Decoder (GSDecoder)

Recently, feed-forward 3DGS methods [6, 12, 96, 97] have enabled fast 3DGS reconstruction from sparse views by training on large datasets, achieving much faster reconstruction than per-scene optimization methods [38]. Leveraging this advantage, our GSDecoder G_ϕ (parameterized by ϕ) is designed to decode pixel-aligned 3DGS from latent representations of K sparse views, given their corresponding camera poses $\mathcal{P} = \{P_i\}_{i=1}^K$ with $P_i = K_i [R_i|T_i]$, where K_i, R_i, and T_i denote the intrinsic matrix, rotation matrix, and translation vector of the i-th view, respectively.

A straightforward approach is to design G_ϕ to directly output the 3D Gaussian parameters from the image latents obtained through the encoder $\mathcal{E}$ as $G_\phi(\{(\mathcal{E}(I_i), P_i)\}_{i=1}^K) = \{(\mu_j, \alpha_j, \Sigma_j, c_j)\}_{j=1}^{H \times W \times K}$, where each 3D Gaussian parameter includes position μ_j, opacity α_j, covariance Σ_j, and color c_j in a pixel-aligned manner with each i-th image $I_i \in \mathbb{R}^{H \times W}$. However, as the image latents are produced through a shared and frozen encoder (to leverage compatibility with 2D generative models during sampling), it might lose fine-grained spatial details, leading to abstract representations. This can limit the preservation of specific 3D structural information. To address these limitations, we propose two improvements to enhance GSDecoder's performance.

Depth latent integration As demonstrated in [37], the encoder $\mathcal{E}$ can effectively encode depth maps. Building on this, we incorporate the depth latents $\{\mathcal{E}(D_i)\}_{i=1}^K$ of each depth map D_i as an additional input to the GSDecoder to further enhance 3D structural information. We use DepthAnythingV2 [112] to extract the depth maps and normalize them to the range [-1, 1], stacking them as three channels similar to RGB for constructing D_i. We observe that this improves the convergence speed and performance of the GSDecoder.

Adversarial loss Applying adversarial losses has demonstrated improvements in the visual quality of decoded output [21, 85]. However, using adversarial loss during training can cause instability, particularly in the early stages. We observe that applying the vision-aided loss [43] after the GSDe-

coder has reached a certain level of convergence significantly improves visual quality without destabilizing training.

Training In summary, our GSDecoder predicts 3DGS $\{(\boldsymbol{\mu}_j, \alpha_j, \boldsymbol{\Sigma}_j, \boldsymbol{c}_j)\}_{j=1}^{H \times W \times K}$ from $\{(\boldsymbol{I}_i, \boldsymbol{D}_i, \boldsymbol{P}_i)\}_{i=1}^{K}$. The model is trained with a combined loss function consisting of LPIPS [123], Mean Squared Error loss, and vision-aided loss [43], computed between the target images of novel views and rendered views from 3DGS. Specifically, the vision-aided loss is activated only after a certain threshold iteration to improve stability and perceptual quality. Additionally, the architecture is initialized with the parameters of the Stable Diffusion 3 decoder, incorporating cross-view attention mechanisms, and the channel dimensions are increased to meet the specific requirements of 3D Gaussian Splatting. Further details on the GSDecoder are provided in Appendix A.

4.3. Multi-View Rectified Flow Model

With the GSDecoder in place, the 3D Gaussian Splatting (3DGS) generation process becomes feasible if we can produce consistent multi-view latents of images $\mathcal{E}(\boldsymbol{I}_i)$, depth maps $\mathcal{E}(\boldsymbol{D}_i)$, and camera poses $\boldsymbol{P}_i$ for $i = 1, \ldots, K$. Our multi-view rectified flow (RF) model is trained to sample from the joint distribution of these latents, conditioned on text C as $p(\{\boldsymbol{I}_i\}_{i=1}^{K}, \{\boldsymbol{D}_i\}_{i=1}^{K}, \{\boldsymbol{P}_i\}_{i=1}^{K} | C)$, allowing for simultaneous generation of multi-view images, depths, and poses. The rationale for modeling a joint distribution instead of each component separately is twofold. First, this approach allows the model to handle various tasks by reformulating them as inpainting problems, as discussed in Section 4.4. Second, joint modeling is crucial for real-world scenes, which require adaptive camera poses tailored to each scene [49].

To achieve this, our multi-view RF model treats the concatenation of multi-view image latents, depth latents, and Plücker ray coordinates [78, 120] along the channel dimension as input data. Formally, each ray $\boldsymbol{r}_i$ is represented as $\langle \boldsymbol{d}_i, \boldsymbol{m}_i \rangle$, where $\boldsymbol{d}_i = \boldsymbol{R}_i^\top \boldsymbol{K}_i^{-1} \boldsymbol{w}_i$ denotes the direction vector and $\boldsymbol{m}_i = (-\boldsymbol{R}_i^\top \boldsymbol{T}_i) \times \boldsymbol{d}_i$ represents the moment vector, with $\boldsymbol{w}_i$ referring to the 2D pixel coordinates. By matching the spatial resolution of $\boldsymbol{w}_i$ with the image and depth latents, each represented as $\mathbb{R}^{n \times h \times w}$, $\boldsymbol{r}_i$ can be expressed as $\mathbb{R}^{6 \times h \times w}$, enabling concatenation along the channel axis.

Through this setup, we define Y_0 as the multi-view latents $\boldsymbol{X}$, where each element $\boldsymbol{X}_i = \langle \mathcal{E}(\boldsymbol{I}_i), \mathcal{E}(\boldsymbol{D}_i), \boldsymbol{r}_i \rangle$ results in $Y_0 = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_K) \in \mathbb{R}^{K \times (2n+6) \times h \times w}$. We then train the multi-view RF model u_θ with the conditional flow matching loss defined in Eq. 4. Specifically, we fine-tune Stable Diffusion 3 [22] by adjusting the channel dimensions in the input and output layers and incorporating cross-view attention. Further details are provided in Appendix A.

Sampling process To recover the camera pose from ray $\boldsymbol{r}_i$, we follow the procedure in [120] and further refine the pose parameters to ensure that multiple views share the same intrinsic. However, we find that naively solving the ODE

can lead to degraded camera pose accuracy. We hypothesize this degradation results from deviations from the camera pose manifold, which introduces errors in camera trajectory estimation. To address this, at each sampling step $t = t_k$, we predict the sampling destination at $t = 0$, recover the camera poses from this prediction, and reconstruct the associated camera rays. These rays are then forwarded to $t = t_k$ to maintain the solution within the valid ray manifold at $t = t_k$.

Stable Diffusion 3 guidance To improve multi-view image generation quality, we integrate vector fields from Stable Diffusion 3, known for robust generalization, with our multi-view RF model to solve the ODE for randomly selected view's image latents at specific sampling steps.

4.4. Editing and Applications via Inpainting

In this section, we demonstrate how SplatFlow can be effectively utilized for 3D editing and applications by applying training-free inversion techniques [16, 62, 67] and inpainting methods [14, 61, 86, 87, 93]. We modify SDEdit [62] inversion for rectified flow-based inversion and adopt the RePaint [61] to support rectified flow inpainting. Additional technical details are provided in Appendix A.

3DGS editing Our GSDecoder is designed to directly decode 3DGS, enabling efficient editing by modifying only the multi-view latents. Additionally, with the RF model's joint modeling of multi-view image latents, edits can be achieved without extra modules (e.g., cross-view attention mechanisms [105, 109]), streamlining the process through simple inversion and sampling. Specifically, given a target text and input multi-view latents, we apply a modified SDEdit for inversion at a chosen timestep t_k. Conditioned on the target text, we then generate the edited multi-view latents by solving the ODE in Eq. 1 following our sampling procedure.

Inpainting application Our model is also applicable to various 3D tasks, as it jointly models multi-view images, depths, and camera poses. This joint modeling allows known parts of the data (e.g., some views or camera parameters) to act as constraints, enabling the prediction of unknown parts through inpainting. In this work, we focus on two specific tasks: 1) *camera pose estimation* from multi-view images and depths, and 2) *novel view synthesis* using a subset of the K multi-view latents and camera poses for novel viewpoints.

5. Experimental Results

In this section, we demonstrate the capabilities of our proposed SplatFlow across 3DGS generation, 3DGS editing, and various 3D tasks. We begin by detailing the implementation of SplatFlow in Section 5.1. Following this, we present comparative results on 3DGS generation in real-world datasets in Section 5.2. In Section 5.3, we evaluate the 3DGS editing performance of SplatFlow against other methods. Finally, Section 5.4 showcases the versatility of our
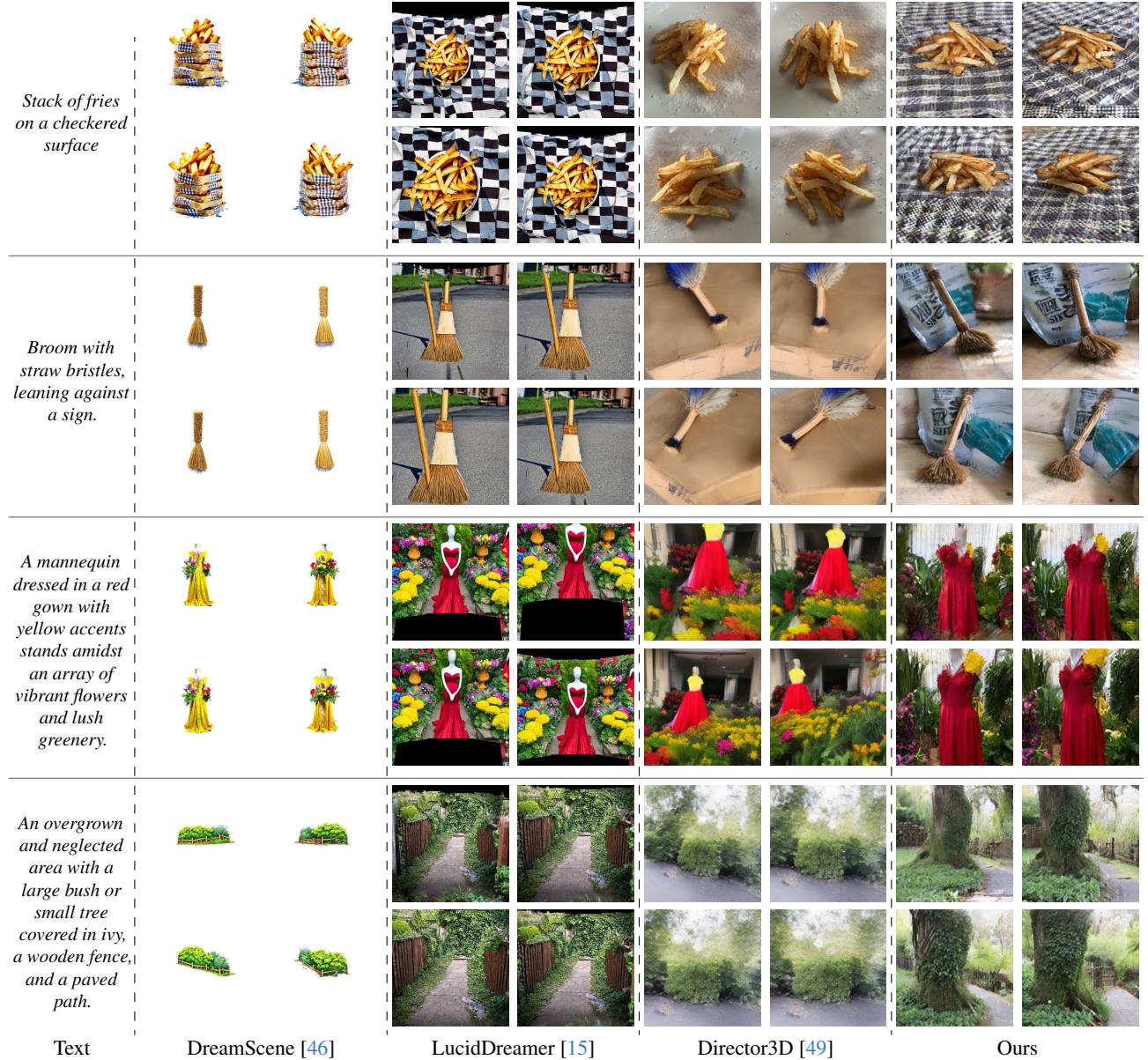
Figure 3. **Qualitative results in text-to-3DGS generation on MVImgNet and DL3DV validation sets.** The first two rows are rendered scenes from the MVImgNet dataset, while the last two rows are from the DL3DV dataset. Our SplatFlow produces cohesive and realistic scenes with sharp details, accurately capturing the intricacies of real-world environments and accommodating diverse camera trajectories.

multi-view RF model in handling diverse 3D tasks through training-free inpainting techniques. Extensive information on all our experiments is provided in Appendix B, with additional results that could not be included here in the Appendix.

5.1. Experimental Setups

Due to space constraints, we provide a concise overview of our experimental setups here. We used the MVImgNet [117] and DL3DV-7K [54] datasets, both containing real-world scenes with images and corresponding camera poses. For evaluation, we utilized 1.25K scenes from MVImgNet for each task and 300 sequences from DL3DV-7K, using the remaining for training. The multi-view RF model and GSDecoder were trained with $K = 8$-view setup. Text descriptions for scenes were extracted using the Llava-One Vision Qwen 7B [45]. We used SDS++ [49] for further improvements.

5.2. Results on Text-to-3DGS Generation

We evaluated our model against various baselines on the DL3DV and MVImgNet validation sets for real-world scene text-to-3DGS generation. Our evaluation protocol utilized the rendered image of the generated 3DGS. We used the FID score [29] to evaluate image quality and the CLIP score [28] to measure alignment with text prompts.

Method	MVImgNet [117]		DL3DV [54]	
	FID-10K↓	CLIPScore↑	FID-2.4K↓	CLIPScore↑
Director3D [49]	39.55	30.48	88.44	30.04
Director3D (w/ SDS++) [49]	41.80	31.00	95.88	31.68
SplatFlow	**34.85**	31.43	**79.91**	30.06
SplatFlow (w/ SDS++)	35.46	**32.30**	85.31	**31.90**

Table 1. **Quantitative results in text-to-3DGS generation on the MVImgNet and DL3DV datasets.** We compared our SplatFlow with and without the SDS++ [49], against Director3D [49].

For quantitative comparison, we adopt Director3D [49], which leverages the full sequences of MVImgNet, DL3DV-10K, and LAION [89] datasets. In contrast, our model was trained on a notably smaller dataset, utilizing only a portion of MVImgNet and DL3DV-7K, without access to the full DL3DV-10K or LAION data. For qualitative results, we also compared against other scene generation methods, including LucidDreamer [15], DreamScene [46], and Director3D [49].

Quantitative results   Table 1 presents the quantitative comparison of SplatFlow and the baseline methods on both MVImgNet and DL3DV datasets. SplatFlow consistently outperforms Director3D [49] in terms of FID and CLIP score, demonstrating its effectiveness in generating high-quality images and maintaining strong alignment with text prompts, even with a smaller training dataset. Specifically, on MVImgNet, SplatFlow achieves a significantly lower FID (34.85 vs. 39.55 for Director3D), indicating better image quality, and a higher CLIP score (31.43 vs. 30.48), showing improved text-to-image alignment. Similarly, on DL3DV, SplatFlow achieves an FID of 79.91, outperforming Director3D's 88.44, with a comparable CLIP score. Notably, with the addition of SDS++ for refinement, both SplatFlow and Director3D show increased CLIP scores, further enhancing text alignment, though at a minor trade-off in FID. This highlights the effectiveness of SDS++ for boosting alignment with text prompts across different 3DGS generation models.

Qualitative results   Figure 3 shows the qualitative comparisons between our method and baselines on DL3DV and MVImgNet validation sets. Overall, SplatFlow demonstrates clear advantages over the baseline methods. First, DreamScene [46], which relies on Score Distillation Sampling (SDS), often struggles to generate realistic real-world scenes, resulting in outputs that lack natural textures and environmental details. Second, LucidDreamer [15], which is based on single-view inpainting, faces challenges when handling scenes with large camera trajectory variations, leading to inconsistent or incomplete views. Third, Director3D [49] tends to produce slightly blurred outputs, lacking the sharpness needed for intricate details. In contrast, our SplatFlow can generate coherent and realistic scenes with clear details, effectively capturing the complexities of real-world environments and various camera trajectories. Additional qualitative results of SplatFlow can be found in Appendix C.
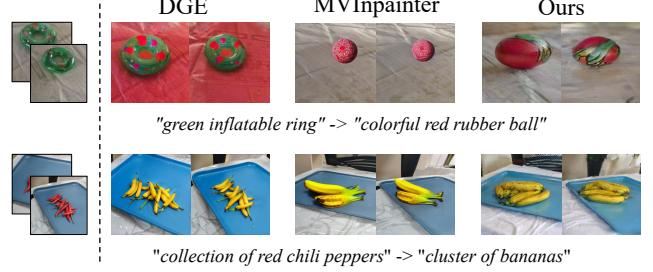

Figure 4. **Qualitative results in 3D editing with MVInpainter [4] and DGE [9].** We show rendered scenes except for MVInpainter.

5.3. Result on 3DGS Editing

We demonstrate the effectiveness of SplatFlow in the 3DGS editing task by creating a benchmark with 100 sampled scenes from the MVImgNet validation set.

Method	CLIPScore↑	CLIP D-sim↑
DGE [9]	27.43	0.102
SplatFlow	28.47	0.169
+) SDS++	**31.30**	**0.224**

Table 2. **3D object replacement.**

Images and texts were provided to GPT-4 to identify main objects and generate text captions for edited images, which were then used for 3D object replacement. For editing, we used the same mask in [4] for masking out the object, then inpainted this area based on the provided editing captions. We compared this pipeline with an optimization-based 3DGS editing method, DGE [9], and conducted qualitative comparisons with MVInpainter [4], which propagates an edited image from pretrained generative models across multiple views. Following standard protocols [11, 34], we measured the CLIP score and directional similarity (CLIP D-sim).

Quantitative results   As shown in Table 2, SplatFlow achieves higher CLIPScore and CLIP D-sim than DGE [9], indicating more accurate and effective editing. SDS++ refinement further boosts performance, demonstrating enhanced alignment with target edits.

Qualitative results   Figure 4 demonstrates that SplatFlow performs comparably to MVInpainter, a specialized 2D inpainting method that leverages an already edited high-quality image. On the other hand, DGE struggles to replace objects fully, often just altering styles instead of achieving complete transformations. This underscores SplatFlow's superior ability in precise 3D object replacement. Please refer to Appendix C for further qualitative results.

5.4. Result on Inpainting Application

Here, we present the adaptability of our multi-view RF model to camera pose estimation and novel-view synthesis via the inpainting technique on the MVImgNet dataset [117].

Camera pose estimation   To demonstrate the effectiveness of our multi-view RF model in camera pose estimation, we primarily compare our approach to RayDiffusion [120] and RelPose++ [51]. We conducted comparisons using an

Method	Rotation↑			Camera Center↑		
	@5	@10	@15	@0.05	@0.1	@0.2
RelPose++ [51]	19.4	37.7	51.4	0.6	12.5	55.0
Ray Regression [120]	10.4	25.6	50.1	15.3	47.9	82.9
Ray Diffusion [120]	17.5	38.7	59.6	24.1	60.9	87.6
SplatFlow (w/ depth)	**26.8**	**52.6**	**59.3**	**62.3**	**91.6**	**99.4**
SplatFlow (w/o depth)	**28.8**	**54.5**	**63.9**	**64.9**	**94.0**	**99.7**

Table 3. **Results in camera pose estimation on MVImgNet validation set.** $@Q$ represents the accuracy threshold for rotations (degrees) and camera centers (units).
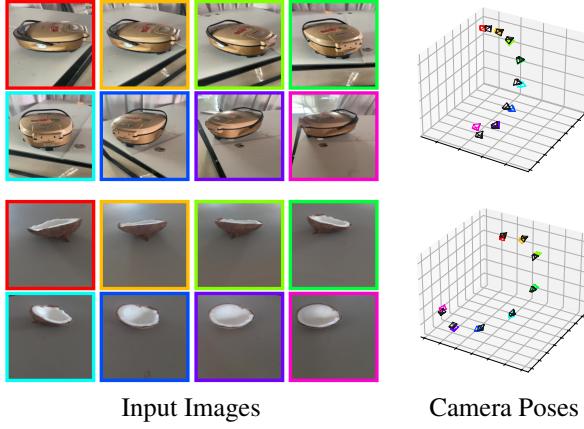

Input Images   Camera Poses

Figure 5. **Qualitative results for camera pose estimation.** Camera poses are estimated from multi-view images. Image border colors match each camera, with black cameras indicating GT poses.

8-view setup, inpainting the ray latent with and without the depth latent from DepthAnythingV2 [112]. We used rotation accuracy$@Q_R$ and translation accuracy$@Q_T$ as evaluation metrics, measuring the proportion of relative camera rotations within Q_R degrees of ground truth and camera centers within Q_T units of scene scale, respectively.

Table 3 and Fig. 5 present the quantitative and qualitative results. Overall, the multi-view RF model achieves superior performance across most scenarios, outperforming the baseline methods. Interestingly, the multi-view RF model achieves higher accuracy without depth latents than with them. We hypothesize that this outcome is due to the joint generation of depth and camera poses, which may yield more detailed and contextually enriched depth estimates.

Novel view-synthesis We evaluated the performance of our multi-view RF model in novel view synthesis under two scenarios: 1) interpolation, with the uniformly sampled input views, and 2) extrapolation, with the input views positioned in the center. This comparison validates our multi-view RF model's capability of 3D reasoning based on relative viewpoints and evaluates its effectiveness in generating novel views. We evaluated PSNR, SSIM, and LPIPS [123] against the ground-truth images for reconstruction quality, as well as absolute relative difference and δ_1 accuracy using depth maps produced by DepthAnythingV2 [112].

Type	RGB			Depth	
	PSNR↑	SSIM↑	LPIPS↓	AbsRel↓	δ_1↑
Interpolation (N=2)	14.73	0.571	0.648	0.588	0.731
Interpolation (N=4)	17.05	0.590	0.551	0.498	0.761
Interpolation (N=6)	18.82	0.626	0.483	0.415	0.775
Extrapolation (N=2)	15.15	0.577	0.627	0.771	0.715
Extrapolation (N=4)	16.80	0.595	0.554	0.679	0.727
Extrapolation (N=6)	17.96	0.613	0.503	0.602	0.747

Table 4. **Novel view synthesis results on MVImgNet.** We use N input views to synthesize $K - N$ novel views, with uniformly sampled views for interpolation and central views for extrapolation.
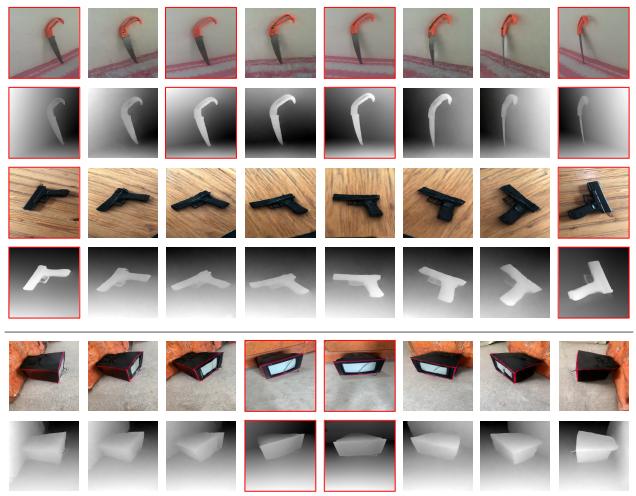

Figure 6. **Qualitative results for novel view synthesis.** Novel view synthesis is performed from the red-box images and depths.

As shown in Fig. 6, our multi-view RF model successfully generates novel views for both scenarios. Also, Table 4 shows that interpolation outperforms extrapolation in both RGB reconstruction and depth estimation as the number of input views increases, indicating that interpolation leverages 3D reasoning for accurate novel view generation, whereas extrapolation focuses on generating diverse novel views.

6. Conclusion

In this paper, we have introduced *SplatFlow*, a novel framework that unifies 3D Gaussian Splatting (3DGS) generation and editing within a single, efficient model. By designing a multi-view Rectified Flow (RF) model alongside a Gaussian Splatting Decoder (GSDecoder), we achieved direct 3DGS generation capable of addressing real-world challenges, such as varying scales and complex camera trajectories. Our approach utilizes joint modeling of multi-view images, depths, and camera poses, enabling seamless integration of training-free inversion and inpainting techniques for a diverse set of 3D tasks. As we present the versatility of our SplatFlow, we believe SplatFlow represents an important step toward a versatile foundational model for 3D content.

References

[1] Yousset I Abdel-Aziz. Direct linear transformation from comparator coordinates into object space coordinates in close range photogrammetry. *Proc. Amer. Soc. Photogrammetry, 1971*, pages 1–19, 1971. 15

[2] Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022. 2, 3

[3] Chong Bao, Yinda Zhang, Bangbang Yang, Tianxing Fan, Zesong Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Sine: Semantic-driven image-based nerf editing with prior-guided editing field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20919–20929, 2023. 1, 3

[4] Chenjie Cao, Chaohui Yu, Yanwei Fu, Fan Wang, and Xiangyang Xue. Mvinpainter: Learning multi-view consistent inpainting to bridge 2d and 3d editing. *arXiv preprint arXiv:2408.08000*, 2024. 7, 17

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 14

[6] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19457–19467, 2024. 2, 4

[7] Anpei Chen, Haofei Xu, Stefano Esposito, Siyu Tang, and Andreas Geiger. Lara: Efficient large-baseline radiance fields. In *European Conference on Computer Vision*, pages 338–355. Springer, 2025. 3

[8] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, et al. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. *arXiv preprint arXiv:2310.00426*, 2023. 2, 3

[9] Minghao Chen, Iro Laina, and Andrea Vedaldi. Dge: Direct gaussian 3d editing by consistent multi-view editing. *arXiv preprint arXiv:2404.18929*, 2024. 2, 7, 17

[10] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 22246–22256, 2023. 18

[11] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21476–21485, 2024. 2, 3, 7

[12] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European Conference on Computer Vision*, pages 370–386. Springer, 2024. 2, 4

[13] Zilong Chen, Feng Wang, Yikai Wang, and Huaping Liu. Text-to-3d using gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21401–21412, 2024. 2, 3

[14] Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022. 2, 5

[15] Jaeyoung Chung, Suyoung Lee, Hyeongjin Nam, Jaerin Lee, and Kyoung Mu Lee. Luciddreamer: Domain-free generation of 3d gaussian splatting scenes. *arXiv preprint arXiv:2311.13384*, 2023. 3, 6, 7

[16] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance. *arXiv preprint arXiv:2210.11427*, 2022. 2, 5

[17] Giannis Daras, Hyungjin Chung, Chieh-Hsin Lai, Yuki Mitsufuji, Jong Chul Ye, Peyman Milanfar, Alexandros G Dimakis, and Mauricio Delbracio. A survey on diffusion models for inverse problems. *arXiv preprint arXiv:2410.00083*, 2024. 2

[18] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 2, 3

[19] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024. 2, 3

[20] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 2

[21] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021. 4

[22] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024. 2, 4, 5, 14, 15

[23] Hyojun Go, Yunsung Lee, Jin-Young Kim, Seunghyun Lee, Myeongho Jeong, Hyun Seung Lee, and Seungtaek Choi. Towards practical plug-and-play diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1962–1971, 2023. 2

[24] Hyojun Go, Yunsung Lee, Seunghyun Lee, Shinhyeok Oh, Hyeongdon Moon, and Seungtaek Choi. Addressing negative transfer in diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024. 2

[25] Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. Diffusion models as plug-and-play priors. *Advances in Neural Information Processing Systems*, 35: 14715–14728, 2022. 2

[26] Yuze He, Yushi Bai, Matthieu Lin, Wang Zhao, Yubin Hu, Jenny Sheng, Ran Yi, Juanzi Li, and Yong-Jin Liu. T^{3}bench: Benchmarking current progress in text-to-3d generation, 2023. 18, 21

[27] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022. 2

[28] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021. 6, 18

[29] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 6, 17

[30] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 15

[31] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2, 3

[32] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. 2

[33] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2room: Extracting textured 3d meshes from 2d text-to-image models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7909–7920, 2023. 3

[34] Hiromichi Kamata, Yuiko Sakuma, Akio Hayakawa, Masato Ishii, and Takuya Narihira. Instruct 3d-to-3d: Text instruction guided 3d-to-3d conversion. *arXiv preprint arXiv:2303.15780*, 2023. 1, 3, 7

[35] Nazmul Karim, Hasan Iqbal, Umar Khalid, Chen Chen, and Jing Hua. Free-editor: zero-shot text-driven 3d scene editing. In *European Conference on Computer Vision*, pages 436–453. Springer, 2025. 1, 3

[36] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in neural information processing systems*, 33:12104–12114, 2020. 14

[37] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9492–9502, 2024. 4

[38] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 2, 3, 4

[39] Umar Khalid, Hasan Iqbal, Nazmul Karim, Jing Hua, and Chen Chen. Latenteditor: Text driven local editing of 3d scenes. *arXiv preprint arXiv:2312.09313*, 2023. 1, 3

[40] Umar Khalid, Hasan Iqbal, Azib Farooq, Jing Hua, and Chen Chen. 3dego: 3d editing on the go! In *European Conference on Computer Vision*, pages 73–89. Springer, 2024. 2, 3

[41] Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2426–2435, 2022. 2

[42] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 15

[43] Nupur Kumari, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Ensembling off-the-shelf models for gan training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10651–10662, 2022. 4, 5, 14

[44] Yunsung Lee, JinYoung Kim, Hyojun Go, Myeongho Jeong, Shinhyeok Oh, and Seungtaek Choi. Multi-architecture multi-expert diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13427–13436, 2024. 2

[45] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024. 6

[46] Haoran Li, Haolin Shi, Wenli Zhang, Wenjun Wu, Yong Liao, Lin Wang, Lik-hang Lee, and Pengyuan Zhou. Dreamscene: 3d gaussian-based text-to-3d scene generation via formation pattern sampling. *arXiv preprint arXiv:2404.03575*, 2024. 6, 7

[47] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343, 2022. 2

[48] Xinyang Li, Zhangyu Lai, Linning Xu, Jianfei Guo, Liujuan Cao, Shengchuan Zhang, Bo Dai, and Rongrong Ji. Dual3d: Efficient and consistent text-to-3d generation with dual-mode multi-view latent diffusion. *arXiv preprint arXiv:2405.09874*, 2024. 15

[49] Xinyang Li, Zhangyu Lai, Linning Xu, Yansong Qu, Liujuan Cao, Shengchuan Zhang, Bo Dai, and Rongrong Ji. Director3d: Real-world camera trajectory and 3d scene generation from text. *Advances in neural information processing systems*, 2024. 3, 5, 6, 7, 18, 20

[50] Zongrui Li, Minghui Hu, Qian Zheng, and Xudong Jiang. Connecting consistency distillation to score distillation for text-to-3d generation. In *European Conference on Computer Vision*, pages 274–291. Springer, 2024. 2

[51] Amy Lin, Jason Y Zhang, Deva Ramanan, and Shubham Tulsiani. Relpose++: Recovering 6d poses from sparse-view observations. *arXiv preprint arXiv:2305.04926*, 2023. 7, 8

[52] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 300–309, 2023. 18

[53] Huan Ling, Seung Wook Kim, Antonio Torralba, Sanja Fidler, and Karsten Kreis. Align your gaussians: Text-to-4d

10

with dynamic 3d gaussians and composed diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8576–8588, 2024. 3

[54] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22160–22169, 2024. 2, 6, 7, 15, 16, 18, 20, 21

[55] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 2, 3

[56] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022. 2, 3

[57] Xian Liu, Xiaohang Zhan, Jiaxiang Tang, Ying Shan, Gang Zeng, Dahua Lin, Xihui Liu, and Ziwei Liu. Humangaussian: Text-driven 3d human generation with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6646–6657, 2024. 2

[58] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. *arXiv preprint arXiv:2309.03453*, 2023. 3

[59] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9970–9980, 2024. 3

[60] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 16

[61] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11461–11471, 2022. 2, 4, 5, 15

[62] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021. 2, 4, 5, 18

[63] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12663–12673, 2023. 18

[64] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2

[65] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on image processing*, 21(12): 4695–4708, 2012. 18

[66] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal processing letters*, 20(3):209–212, 2012. 18

[67] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6038–6047, 2023. 2, 5

[68] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 1

[69] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021. 2

[70] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022. 2

[71] Francesco Palandra, Andrea Sanchietti, Daniele Baieri, and Emanuele Rodolà. Gsedit: Efficient text-guided editing of 3d objects via gaussian splatting. *arXiv preprint arXiv:2403.05154*, 2024. 2, 3

[72] Byeongjun Park and Changick Kim. Point-dynrf: Point-based dynamic radiance fields from a monocular video. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3171–3181, 2024. 3

[73] Byeongjun Park, Sangmin Woo, Hyojun Go, Jin-Young Kim, and Changick Kim. Denoising task routing for diffusion models. *arXiv preprint arXiv:2310.07138*, 2023. 2

[74] Byeongjun Park, Hyojun Go, and Changick Kim. Bridging implicit and explicit geometric transformation for single-image view synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 2

[75] Byeongjun Park, Hyojun Go, Jin-Young Kim, Sangmin Woo, Seokil Ham, and Changick Kim. Switch diffusion transformer: Synergizing denoising tasks with sparse mixture-of-experts. *arXiv preprint arXiv:2403.09176*, 2024. 2

[76] Jangho Park, Gihyun Kwon, and Jong Chul Ye. Ed-nerf: Efficient text-guided editing of 3d scene using latent space nerf. *arXiv preprint arXiv:2310.02712*, 2023. 1, 3

[77] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11410–11420, 2022. 16

[78] Julius Plücker. *Analytisch-geometrische Entwicklungen*. GD Baedeker, 1828. 5

[79] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023. 2, 3

[80] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 2, 18

[81] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 14

[82] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10901–10911, 2021. 20

[83] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. In *ACM SIGGRAPH 2023 conference proceedings*, pages 1–11, 2023. 3

[84] Robin Rombach, Patrick Esser, and Björn Ommer. Geometry-free view synthesis: Transformers and no 3d priors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14356–14366, 2021. 2

[85] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 2, 3, 4, 14

[86] Litu Rout, Advait Parulekar, Constantine Caramanis, and Sanjay Shakkottai. A theoretical justification for image inpainting using denoising diffusion probabilistic models. *arXiv preprint arXiv:2302.01217*, 2023. 2, 5

[87] Litu Rout, Negin Raoof, Giannis Daras, Constantine Caramanis, Alex Dimakis, and Sanjay Shakkottai. Solving linear inverse problems provably via posterior sampling with latent diffusion models. *Advances in Neural Information Processing Systems*, 36, 2023. 2, 5

[88] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, pages 87–103. Springer, 2024. 2

[89] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022. 3, 7, 20

[90] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023. 3

[91] Jaidev Shriram, Alex Trevithick, Lingjie Liu, and Ravi Ramamoorthi. Realmdreamer: Text-driven 3d scene generation with inpainting and depth diffusion. *arXiv preprint arXiv:2404.07199*, 2024. 3

[92] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. 2, 3

[93] Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2023. 2, 5

[94] Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning*, pages 32483–32498. PMLR, 2023. 2

[95] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 2, 3

[96] Stanislaw Szymanowicz, Chrisitian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10208–10217, 2024. 4

[97] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In *European Conference on Computer Vision*, pages 1–18. Springer, 2024. 2, 3, 4

[98] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In *The Twelfth International Conference on Learning Representations*, 2024. 2, 3

[99] Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, Karsten Kreis, et al. Lion: Latent point diffusion models for 3d shape generation. *Advances in Neural Information Processing Systems*, 35:10021–10039, 2022. 2

[100] Alexander Vilesov, Pradyumna Chari, and Achuta Kadambi. Cg3d: Compositional generation for text-to-3d via gaussian splatting. *arXiv preprint arXiv:2311.17907*, 2023. 2

[101] Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitry Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. Sv3d: Novel multi-view synthesis and 3d generation from a single image using latent video diffusion. In *European Conference on Computer Vision*, pages 439–457. Springer, 2025. 3

[102] Can Wang, Mingming He, Menglei Chai, Dongdong Chen, and Jing Liao. Mesh-guided neural implicit field editing. *arXiv preprint arXiv:2312.02157*, 2023. 3

[103] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12619–12629, 2023. 18

[104] Junjie Wang, Jiemin Fang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. Gaussianeditor: Editing 3d gaussians delicately with text instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20902–20911, 2024. 2, 3

[105] Yuxuan Wang, Xuanyu Yi, Zike Wu, Na Zhao, Long Chen, and Hanwang Zhang. View-consistent 3d editing with gaussian splatting. In *European Conference on Computer Vision*, pages 404–420. Springer, 2024. 2, 3, 5

[106] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 17

[107] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36, 2024. 18

[108] Sangmin Woo, Byeongjun Park, Hyojun Go, Jin-Young Kim, and Changick Kim. Harmonyview: Harmonizing consistency and diversity in one-image-to-3d. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10574–10584, 2024. 3

[109] Jing Wu, Jia-Wang Bian, Xinghui Li, Guangrun Wang, Ian Reid, Philip Torr, and Victor Adrian Prisacariu. Gaussctrl: multi-view consistent text-driven 3d gaussian splatting editing. *arXiv preprint arXiv:2403.08733*, 2024. 2, 3, 5

[110] Hanyuan Xiao, Yingshu Chen, Huajian Huang, Haolin Xiong, Jing Yang, Pratusha Prasad, and Yajie Zhao. Localized gaussian splatting editing with contextual awareness. *arXiv preprint arXiv:2408.00083*, 2024. 2, 3

[111] Yinghao Xu, Hao Tan, Fujun Luan, Sai Bi, Peng Wang, Jiahao Li, Zifan Shi, Kalyan Sunkavalli, Gordon Wetzstein, Zexiang Xu, and Kai Zhang. DMV3d: Denoising multi-view diffusion using 3d large reconstruction model. In *The Twelfth International Conference on Learning Representations*, 2024. 2

[112] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv preprint arXiv:2406.09414*, 2024. 4, 8, 16, 19

[113] Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6796–6807, 2024. 2

[114] Xuanyu Yi, Zike Wu, Qingshan Xu, Pan Zhou, Joo-Hwee Lim, and Hanwang Zhang. Diffusion time-step curriculum for one image to 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9948–9958, 2024. 3

[115] Lu Yu, Wei Xiang, and Kang Han. Edit-diffnerf: Editing 3d neural radiance fields using 2d diffusion model. *arXiv preprint arXiv:2306.09551*, 2023. 1, 3

[116] Wangbo Yu, Jinbo Xing, Li Yuan, Wenbo Hu, Xiaoyu Li, Zhipeng Huang, Xiangjun Gao, Tien-Tsin Wong, Ying Shan, and Yonghong Tian. Viewcrafter: Taming video diffusion models for high-fidelity novel view synthesis. *arXiv preprint arXiv:2409.02048*, 2024. 2

[117] Xianggang Yu, Mutian Xu, Yidan Zhang, Haolin Liu, Chongjie Ye, Yushuang Wu, Zizheng Yan, Chenming Zhu, Zhangyang Xiong, Tianyou Liang, et al. Mvimgnet: A large-scale dataset of multi-view images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9150–9161, 2023. 2, 6, 7, 15, 16, 18, 20, 21

[118] Bowen Zhang, Yiji Cheng, Jiaolong Yang, Chunyu Wang, Feng Zhao, Yansong Tang, Dong Chen, and Baining Guo. Gaussiancube: Structuring gaussian splatting using optimal transport for 3d generative modeling. *arXiv preprint arXiv:2403.19655*, 2024. 2

[119] Jingbo Zhang, Xiaoyu Li, Ziyu Wan, Can Wang, and Jing Liao. Text2nerf: Text-driven 3d scene generation with neural radiance fields. *IEEE Transactions on Visualization & Computer Graphics*, 30(12):7749–7762, 2024. 1

[120] Jason Y. Zhang, Amy Lin, Moneish Kumar, Tzu-Hsuan Yang, Deva Ramanan, and Shubham Tulsiani. Cameras as rays: Pose estimation via ray diffusion. In *The Twelfth International Conference on Learning Representations*, 2024. 2, 5, 7, 8, 15

[121] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *European Conference on Computer Vision*, pages 1–19. Springer, 2024. 2

[122] Kaiyi Zhang, Yang Chen, Ximing Yang, Weizhong Zhang, and Cheng Jin. Point cloud part editing: Segmentation, generation, assembly, and selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7187–7195, 2024. 3

[123] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 5, 8, 14, 17

[124] Yudi Zhang, Qi Xu, and Lei Zhang. Dragtex: Generative point-based texture editing on 3d mesh. *arXiv preprint arXiv:2403.02217*, 2024. 3

[125] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5826–5835, 2021. 2

[126] Shijie Zhou, Zhiwen Fan, Dejia Xu, Haoran Chang, Pradyumna Chari, Tejas Bharadwaj, Suya You, Zhangyang Wang, and Achuta Kadambi. Dreamscene360: Unconstrained text-to-3d scene generation with panoramic gaussian splatting. In *European Conference on Computer Vision*, pages 324–342. Springer, 2025. 3

[127] Jingyu Zhuang, Di Kang, Yan-Pei Cao, Guanbin Li, Liang Lin, and Ying Shan. Tip-editor: An accurate 3d editor following both text-prompts and image-prompts. *ACM Transactions on Graphics (TOG)*, 43(4):1–12, 2024. 2, 3

A. Methodology Details

In this section, we provide additional details about the methodology of our proposed SplatFlow model, extending the description given in Section 4. We begin by elaborating on the architecture and training details of the Gaussian Splatting Decoder (GSDecoder) in Section A.1, covering the specific modifications made to adapt the Stable Diffusion 3 decoder to 3D Gaussian Splatting (3DGS). Following that, we provide details on the Multi-View Rectified Flow (RF) model, including the architecture, loss functions, and several modifications made to the sampling process to jointly generate multi-view images, depths, and camera poses in Section A.2. Finally, we describe the editing process employed by SplatFlow, discussing how training-free inversion and inpainting techniques are applied to facilitate seamless 3D editing in Section A.3.

A.1. Gaussian Splatting Decoder (GSDecoder)

Architecture   Our GSDecoder architecture builds upon the Stable Diffusion 3 decoder architecture [22], with key modifications to adapt it for 3D Gaussian Splatting (3DGS). Specifically, we adjusted the input channel size to accommodate the concatenated latents of images, depths, and rays, and altered the output channel size to produce pixel-aligned 3DGS parameters. Furthermore, we modified the attention layers to incorporate cross-view attention, enabling the attention mechanism to operate across all view tokens simultaneously, rather than processing tokens for each view independently. We initialized the GSDecoder weights using the pre-trained weights from the Stable Diffusion decoder for all layers, except for the input and output layers. For these layers, we initialized the first channels with the corresponding Stable Diffusion weights, and the remaining channels were initialized by copying these values.

Loss function   Our GSDecoder is trained using a weighted sum of three losses: mean squared error loss, LPIPS [123], and vision-aided GAN loss [43]. Specifically, the vision-aided GAN loss leverages backbones from DINO [5] and CLIP [81], and we incorporated differentiable augmentation [36] along with a multi-level version of hinge loss[1]. Therefore, our loss can be represented as:

$$L_{\text{decoder}} = w_1 L_{\text{mse}} + w_2 L_{\text{LPIPS}} + w_3 L_{\text{vision-aided}}, \quad (5)$$

where L_{mse}, L_{LPIPS}, and $L_{\text{vision-aided}}$ represent the mean squared error loss, LPIPS loss, and vision-aided GAN loss, respectively, all computed between the rendered images from the 3DGS and the target view images. The weight factors for each loss are denoted by w_1, w_2, and w_3, and We set $w_1 = 1$ and $w_2 = 0.05$. Regarding w_3, we turn it after training undergoes specific iterations, and we utilize the adaptive weighting scheme in [85]. Specifically, w_3 is determined

---

[1]https://github.com/nupurkmr9/vision-aided-gan

Algorithm 1 Sampling Process of *SplatFlow*

Input:
- $\boldsymbol{u}_\theta$ ▷ *Velocity function of SplatFlow*
- $\boldsymbol{v}_\phi$ ▷ *Velocity function of SD3*
- $t = [t_N, \ldots, t_0]$ ▷ *Timesteps*
- t_{stop} ▷ *A time step to stop updating ray latents*
- $\boldsymbol{Y}_{t_N} = \boldsymbol{X}_{t_N}^{(1:K)} = (\boldsymbol{X}_{t_N}^1 \ldots \boldsymbol{X}_{t_N}^K) \sim \mathcal{N}(0, I)$ ▷ *Initial Noise*

Sampling:
1: **for** $i = N, \ldots, 1$ **do**
2:    $\hat{\boldsymbol{v}}_{t_i} \leftarrow \boldsymbol{u}_\theta(\boldsymbol{Y}_{t_i}, t_i)$
3:    **if** $i \geq t_{stop}$ **then**
4:        **if** $N - i \equiv 0 \mod 3 \ \& \ i \neq t_{stop}$ **then**
5:            $\hat{\boldsymbol{v}}_{t_i}[: n] \leftarrow \boldsymbol{v}_\phi(\boldsymbol{Y}_{t_i}[: n], t_i)$ ▷ *Replace to SD3*
6:        **end if**
7:        $\tilde{\boldsymbol{Y}}_{t_0} \leftarrow \boldsymbol{Y}_{t_i} - t_i \boldsymbol{u}_\theta(\boldsymbol{Y}_{t_i}, t_i)$ ▷ *Predict Destination*
8:        $\langle \boldsymbol{d}^{(1:K)}, \boldsymbol{m}^{(1:K)} \rangle \leftarrow \tilde{\boldsymbol{Y}}_{t_0}[2n :]$ ▷ *Extract Ray Latent*
9:        **for** $j = 1, \ldots, K$ **do**
10:           $\langle \boldsymbol{K}^j, \boldsymbol{R}^j, \boldsymbol{T}^j \rangle \leftarrow \text{ray\_optimize}(\langle \boldsymbol{d}^j, \boldsymbol{m}^j \rangle)$
11:       **end for**
12:       $\langle \boldsymbol{K}, \boldsymbol{R}^{(1:K)}, \boldsymbol{T}^{(1:K)} \rangle \leftarrow \text{shared\_K}(\langle \boldsymbol{K}, \boldsymbol{R}, \boldsymbol{T} \rangle^{(1:K)})$
13:       $\langle \boldsymbol{d}^{(1:K)}, \boldsymbol{m}^{(1:K)} \rangle \leftarrow \text{plücker}(\langle \boldsymbol{K}, \boldsymbol{R}^{(1:K)}, \boldsymbol{T}^{(1:K)} \rangle)$
14:       $\boldsymbol{r}_{t_0} \leftarrow \langle \boldsymbol{d}^{(1:K)}, \boldsymbol{m}^{(1:K)} \rangle$ ▷ *Update Ray Destination*
15:   **end if**
16:   $\boldsymbol{Y}_{t_{i-1}} \leftarrow \boldsymbol{Y}_{t_i} + (t_{i-1} - t_i) \hat{\boldsymbol{v}}_{t_i}(\boldsymbol{Y}_{t_i}, t_i)$
17:   $\boldsymbol{z} \sim \mathcal{N}(0, I)$
18:   $\boldsymbol{Y}_{t_{i-1}}[2n :] \leftarrow (1 - t_{i-1}) \boldsymbol{r}_{t_0} + t_{i-1} \boldsymbol{z}$
19: **end for**
20: **return** $\boldsymbol{Y}_{t_0}, \boldsymbol{K}, \boldsymbol{R}^{(1:K)}, \boldsymbol{T}^{(1:K)}$

---

at each training iteration based on the ratio of l_2-norm of the gradient of other loss functions to the gradient of the vision-aided GAN loss at the last layer parameters of the GSDecoder. This ratio is then multiplied by 0.1 to set w_3.

A.2. Multi-View Rectified Flow Model

Architecture   The architecture of our multi-view rectified flow (RF) model is primarily based on the Stable Diffusion 3 medium [22][2], with modifications to fit our requirements. We expanded the input and output channels to accommodate concatenated latents, and updated the attention mechanism to incorporate cross-view attention. The model was initialized using pre-trained weights from Stable Diffusion 3. For the input and output layers, the extra channels were initialized by copying the pre-trained weights.

Training   Following practices in Stable Diffusion 3 [22], we applied an SNR sampler and used three text encoders.

---

[2]https://huggingface.co/stabilityai/stable-diffusion-3-medium

Sampling Process Since our multi-view RF model generates images, depths, and camera poses simultaneously, we modified the sampling process to effectively handle these joint tasks. Algorithm 1 outlines our sampling procedure, emphasizing three key modifications:

- **Early stopping of camera pose updates**: We adopt the early stopping approach from RayDiffusion [120], where camera poses are determined early in the sampling process and remain fixed for subsequent steps. This prevents instability and helps maintain a consistent reference frame for the generated views.
- **Intermediate pose optimization with constrained manifold**: To improve camera pose estimation, we introduce a step to predict the sampling destination for ray latents, regress camera poses, and project the resulting Plücker ray representation onto a valid ray manifold at each sampling step. This helps avoid error accumulation and ensures that the poses remain accurate throughout the entire process.
- **Stable Diffusion 3 guidance for generalization**: We integrate vector fields from Stable Diffusion 3 [22] into the sampling process before fixing the camera poses. This enhances the generalizability of our model, especially given the smaller in-the-wild 3D scene datasets we use [54, 117]. Applying this guidance early ensures consistency between multi-view images and depths while improving their quality. Also, we use the dual-mode toggling approach similar to Dual3D [48], applying the guidance every three sampling steps to balance generalizability with 3D consistency.

To regress the camera poses for j-th view $\boldsymbol{K}^j, \boldsymbol{R}^j, \boldsymbol{T}^j$ from the Plücker ray representation with $h \times w$ rays, we use the same optimization process in RayDiffusion [120] as:

$$\boldsymbol{c}^j = \underset{\boldsymbol{p} \in \mathbb{R}^3}{\arg\min} \sum_{\langle \boldsymbol{d}^j, \boldsymbol{m}^j \rangle \in \mathbb{R}} \|\boldsymbol{p} \times \boldsymbol{d}^j - \boldsymbol{m}^j\|^2, \quad (6)$$

$$\boldsymbol{P}^j = \underset{\|\boldsymbol{H}\|=1}{\arg\min} \sum_{i=1}^{h \times w} \|\boldsymbol{H}\boldsymbol{d}_i^j \times \boldsymbol{u}_i\|, \quad (7)$$

where $\boldsymbol{u}$ is the per-pixel ray directions of an identity camera (*i.e.*, $\boldsymbol{K} = \boldsymbol{I}$ and $\boldsymbol{R} = \boldsymbol{I}$). Then, the projection matrix $\boldsymbol{P}^j$ is decomposed into the intrinsic matrix $\boldsymbol{K}^j$ and the rotation matrix $\boldsymbol{R}^j$ via DLT [1]. We further optimize intrinsic and rotation matrices for K views using an Adam [42] optimizer with 10 iterations (which adds negligible overhead), ensuring that all views share the same intrinsic matrix as:

$$\langle \boldsymbol{K}, \boldsymbol{R}^{(1:K)} \rangle = \underset{\|\boldsymbol{R}\|=1}{\arg\min} \sum_{j=1}^{K} \sum_{i=1}^{h \times w} \|\boldsymbol{R}^j \boldsymbol{d}_i^j \times \boldsymbol{u}_{K,i}\|, \quad (8)$$

where $\boldsymbol{u}_K$ is the per-pixel ray directions of a camera with the intrinsic matrix $\boldsymbol{K}$ and the identity rotation matrix. Then, we calculate the translation vector of each view $\boldsymbol{T}^j = -\boldsymbol{R}^{j^\top} \boldsymbol{c}^j$. We set the total sampling steps N to 200 and t_{stop} to 150. We

---

Algorithm 2 Inpainting Process of *SplatFlow*

Input:

$\boldsymbol{u}_\theta$ ▷ *Velocity function of SplatFlow*
$t = [t_N, \ldots, t_0]$ ▷ *Timesteps*
t_{stop} ▷ *A timestep to stop updating ray latents*
$Y_{t_0}^{known}$ ▷ *Known latents*
m ▷ *Mask for known latents*
$\boldsymbol{Y}_{t_N} = \boldsymbol{X}_{t_N}^{(1:K)} = (\boldsymbol{X}_{t_N}^1 \ldots \boldsymbol{X}_{t_N}^K) \sim \mathcal{N}(0, I)$ ▷ *Initial noise*

Sampling:

1: **for** $i = N, \ldots, 1$ **do**
2:    **if** $i \geq t_{stop}$ **then**
3:      $\tilde{\boldsymbol{Y}}_{t_0} \leftarrow \boldsymbol{Y}_{t_i} - t_i \boldsymbol{u}_\theta(\boldsymbol{Y}_{t_i}, t_i)$ ▷ *Predict Destination*
4:      $\langle \boldsymbol{d}^{(1:K)}, \boldsymbol{m}^{(1:K)} \rangle \leftarrow \tilde{\boldsymbol{Y}}_{t_0}[2n :]$ ▷ *Extract Ray Latent*
5:      **for** $j = 1, \ldots, K$ **do**
6:        $\langle \boldsymbol{K}^j, \boldsymbol{R}^j, \boldsymbol{T}^j \rangle \leftarrow \text{ray\_optimize}(\langle \boldsymbol{d}^j, \boldsymbol{m}^j \rangle)$
7:      **end for**
8:      $\langle \boldsymbol{K}, \boldsymbol{R}^{(1:K)}, \boldsymbol{T}^{(1:K)} \rangle \leftarrow \text{shared\_K}(\langle \boldsymbol{K}, \boldsymbol{R}, \boldsymbol{T} \rangle^{(1:K)})$
9:      $\langle \boldsymbol{d}^{(1:K)}, \boldsymbol{m}^{(1:K)} \rangle \leftarrow \text{plücker}(\langle \boldsymbol{K}, \boldsymbol{R}^{(1:K)}, \boldsymbol{T}^{(1:K)} \rangle)$
10:     $\boldsymbol{r}_{t_0} \leftarrow \langle \boldsymbol{d}^{(1:K)}, \boldsymbol{m}^{(1:K)} \rangle$ ▷ *Update Ray Destination*
11:    **end if**
12:    $\boldsymbol{Y}_{t_{i-1}}^{\text{unknown}} \leftarrow \boldsymbol{Y}_{t_i} + (t_{i-1} - t_i) \boldsymbol{u}_\theta(\boldsymbol{Y}_{t_i}, t_i)(\boldsymbol{Y}_{t_i}, t_i)$
13:    $\boldsymbol{z} \sim \mathcal{N}(0, I)$
14:    $\boldsymbol{Y}_{t_{i-1}}^{\text{unknown}}[2n :] \leftarrow (1 - t_{i-1})\boldsymbol{r}_{t_0} + t_{i-1}\boldsymbol{z}$
15:    $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$
16:    $Y_{t_{i-1}}^{\text{known}} = (1 - t_{i-1})Y_{t_0}^{\text{known}} + t_{i-1}\boldsymbol{\epsilon}$
17:    $Y_{t_{i-1}} = m \odot Y_{t_{i-1}}^{\text{known}} + (1 - m) \odot Y_{t_{i-1}}^{\text{unknown}}$
18: **end for**
19: **return** $\boldsymbol{Y}_{t_0}, \boldsymbol{K}, \boldsymbol{R}^{(1:K)}, \boldsymbol{T}^{(1:K)}$

---

employ different classifier-free guidance [30] scales to solve ODE for each latent, where we use 7, 5, and 1 for image, depth, and ray latents, respectively. We set the classifier-free guidance scale to 3 for Stable Diffusion 3 guidance.

A.3. Inpainting Process

By integrating the RePaint [61] into the rectified flow model, our multi-view RF model becomes adaptable to 3DGS editing and training-free downstream tasks such as 3D object replacement, novel view synthesis, and camera pose estimation. Algorithm 2 provides an overview of our inpainting process, which incorporates an early stopping strategy and intermediate ray inversion during the sampling process. Furthermore, we utilize RePaint [61] by leveraging the inversion of known latents to refine the denoised unknown latents at the final stage of each sampling step. As the inpainting process depends on conditions derived from known latents, we exclude the use of Stable Diffusion 3 guidance during the inpainting process.

B. Experimental Setup Details

In this section, we provide comprehensive details regarding the experimental setups used in our paper, extending beyond the brief description given in Section 5.

B.1. Implementation Details

Dataset   As described in [117], the MVImgNet dataset consists of 219,188 scenes annotated with camera parameters. After removing erroneous scenes, we retained approximately 210K scenes. From these, 10K scenes were allocated as the validation set, with 1.25K scenes designated for the validation of each specific task. The rationale for sampling is that fully evaluating all 10K scenes is computationally intensive. The DL3DV [54] dataset originally contained 10K scenes, but during our experimental period, only 7K scenes were available. Therefore, we utilized the 7K available sequences and allocated 300 sequences as the validation set. Since neither dataset includes text annotations for each scene, we extracted text descriptions by utilizing the Llava-One Vision Qwen7B model. A random image was selected to generate the corresponding text descriptions.

Training configuration   Excluding the validation set described above, we used the remaining dataset to train Splat-Flow. Both the GSDecoder and the multi-view RF model were trained with an 8-view setup, sampling 8 viewpoints per scene. Specifically:

- **GSDecoder:** Trained for 400K iterations with a batch size of 8, using the AdamW optimizer [60] and a learning rate of 5×10^{-5}. The vision-aided GAN loss was activated at 200K iterations, during which the discriminator learning rate was doubled to 1×10^{-4}. For depth estimation, we used the DepthAnythingV2 Small model [112][3].
- **Multi-view RF model:** Trained for 100K iterations with a batch size of 256, using the AdamW optimizer with a learning rate of 1×10^{-4}. The learning rate was linearly warmed up for the initial 1K steps. For extracting the depth map, we used the DepthAnythingV2 Small model as in the GSDecoder.

B.2. Detailed Setups in Text-to-3DGS Generation

Evaluation protocol   We evaluated the performance of our 3DGS generation model using text annotations from the validation sets of the MVImgNet and DL3DV datasets. The corresponding ground-truth images were used as reference images for calculating Fréchet Inception Distance (FID) and CLIP scores. Specifically, we used 10K reference images from MVImgNet and 2.4K reference images from DL3DV.

FID score was calculated using CleanFID [77][4] to assess the distance between the generated and reference im-

ages. CLIP scores were computed using the "openai/clip-vit-base-patch16" model to measure the alignment between the generated images and text descriptions. These quantitative measurements are conducted for the rendered 8-views.

B.3. Detailed Setups in 3DGS Editing

Evaluation protocol   We conducted a benchmark on 100 scenes from the MVImgNet dataset to evaluate object replacement capabilities. For this, we used GPT-4o with the following prompt:

```
You are a vision-language model designed to
    create captions that describe object
    replacements in images. Given an input
    image and its caption, your task is to
    produce a new caption where the primary
    object is replaced with a different one,
    maintaining the overall context and
    scene.

Guidelines:
1. Object Replacement Focus: Change the
   main object in the caption to a
   different but plausible one for the
   scene. Keep other details consistent
   with the original setting (e.g.,
   background, lighting).

2. Natural Integration: Ensure the new
   object fits logically within the
   environment described. Avoid improbable
   replacements that clash with the scene's
    context or elements.

3. Clarity and Directness: Use clear and
   straightforward language to describe the
    new object in place of the original,
   reflecting the same style as the given
   caption.

4. Single Object Focus: Most images will
   contain a single object, so focus solely
    on replacing that object without
   altering other aspects of the scene
   unless explicitly instructed.

5. If the given caption describe empty
   scene like empty hallway, add a 'new
   object' to the scene.

6. Just return the text.

Example:

- Input Caption: "A white tag with a green
   leaf design and the text \"HEY!\" on it
   .|Leaf-shaped tag on hanger, black and
   white checkered background."
```

Method	PSNR↑	LPIPS↓	SSIM↑	FID-50K↓
w/o Depth Latent (200K Iterations)	25.64	0.2507	0.7993	16.29
w/ Depth Latent (200K Iterations)	26.19	0.2260	0.8169	11.92
w/ Depth Latent (400K Iterations)	26.68	0.2129	0.8251	8.80
+ Vision-Aided GAN Loss	**26.84**	**0.2048**	**0.8256**	**5.81**

Table 5. **Ablation study on GSDecoder design choices.** Evaluations are performed using PSNR, LPIPS, SSIM, and FID, highlighting the impact of incorporating depth latents and vision-aided GAN loss in improving 3DGS quality.

```
– Target Caption: "A sleek metallic spoon
  with a reflective surface on a plaid
  fabric."
```

Out of 100 scenes, GPT-4o successfully generated captions for 98 scenes. Two scenes failed due to response refusal from GPT-4o. Evaluation metrics were then calculated on 8 rendered views per method, using the newly generated captions to guide editing.

Comparison Methods. We used DGE [9] and MVInpainter [4] as baselines by using their official implementations with the following configurations:

- **DGE** [9]: To create 3D Gaussian Splatting (3DGS) representations as the initial point for DGE, all viewpoint images were utilized. Subsequently, 3DGS editing was performed using the provided captions.
- **MVInpainter** [4]: Similar to our method, MVInpainter extracts 8 views and generates masks for these views. These masks are then used with Stable Diffusion 2, a text-to-image inpainting model, to edit the first view. The remaining views are inpainted based on the edited primary view.

C. Additional Experimental Results

To comprehensively validate the effectiveness of SplatFlow, we provide additional experimental results in this section.

C.1. Ablation on GSDecoder Design Choice

To validate the effectiveness of our design choices in the GSDecoder, we conducted ablation studies focusing on two key aspects: (1) the incorporation of depth latents, and (2) the impact of the vision-aided GAN loss. We analyzed these effects by comparing four variants of our GSDecoder:

- **Without Depth Latents (200K iterations):** A baseline variant that excludes depth latents during training to evaluate the effect of incorporating depth information.
- **With Depth Latents (200K iterations):** This version includes depth latents to assess their contribution to improving the quality of the generated 3D Gaussian Splatting.
- **With Depth Latents (400K iterations):** We extended the training by 200K iterations to examine the impact of prolonged training without the vision-aided GAN loss.
- **With Depth Latents + Vision-Aided GAN Loss (400K iterations):** This variant applies the vision-aided GAN

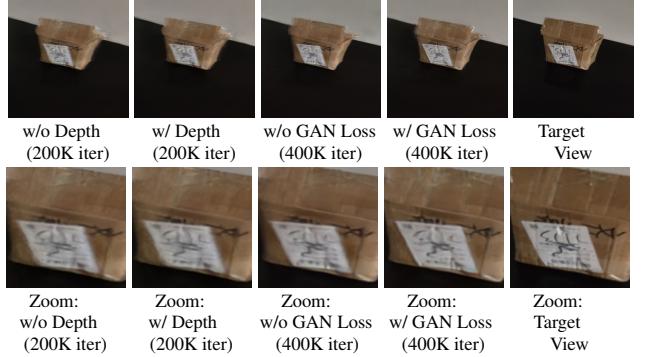| w/o Depth (200K iter)	w/ Depth (200K iter)	w/o GAN Loss (400K iter)	w/ GAN Loss (400K iter)	Target View
Zoom: w/o Depth (200K iter)	Zoom: w/ Depth (200K iter)	Zoom: w/o GAN Loss (400K iter)	Zoom: w/ GAN Loss (400K iter)	Zoom: Target View

Figure 7. **Ablation study on GSDecoder design choices.** The first row shows the original views, while the second row provides zoomed-in details for better visualization. Incorporating depth latents and vision-aided GAN loss enhances the realism and quality of generated 3D Gaussian Splatting (3DGS) scenes.

loss starting after 200K iterations to evaluate the impact of adversarial training on enhancing 3DGS quality.

For training, we utilized the MVImgNet dataset excluding the 10K validation split. We evaluated the generated outputs using 5 rendered images per scene, measured by PSNR, LPIPS [123], SSIM [106], and FID [29]. For FID calculations, we sampled 50K reference images.

The results of this ablation study are illustrated in Table 5. As shown, incorporating depth latents and vision-aided GAN loss both contributed significantly to improving the quality of 3DGS. 1) Depth latents: Incorporating depth latents led to substantial improvements in PSNR, LPIPS, SSIM, and FID metrics compared to the baseline without depth information. This demonstrates that including depth information enhances the quality and consistency of the decoded scenes. 2) Vision-aided GAN loss: Adding vision-aided GAN loss after 200K iterations yielded the best performance across all metrics. Compared to training without vision-aided GAN loss, it significantly improved perceptual quality, as evidenced by better LPIPS and FID scores.

Additionally, the qualitative comparison of the four GS-Decoder variants is presented in Fig. 7. The images demonstrate that incorporating depth latents significantly enhances the sharpness and detail of the generated scenes compared to the baseline without depth information, leading to more accurate reconstructions of the target view. Furthermore, adding the vision-aided GAN loss at 400K iterations results in the most visually compelling outputs, with enhanced texture details and consistency across views. This progression from the baseline to the final variant clearly highlights the positive impact of both depth information and adversarial training on the quality of novel view synthesis. Specifically, the final configuration (w/ GAN Loss, 400K iter) shows improvements in fine-grained textures and overall coherence, making it visually closer to the target view.

Method	FID-10K↓	CLIPScore↑
Stop-Ray ($t_{\text{stop}} = 100$)	35.55	31.37
Stop-Ray ($t_{\text{stop}} = 50$)	37.89	31.41
w/o Stop-Ray ($t_{\text{stop}} = 0$)	47.32	30.12
SplatFlow - Default ($t_{\text{stop}} = 150$)	**34.85**	**31.43**

Table 6. **Impact of the Stop-Ray modification.** Evaluations are conducted using FID-10K and CLIPScore metrics to assess the effectiveness of stopping camera ray updates at different timesteps in the sampling process.

Method	FID-10K↓	CLIPScore↑
SplatFlow (w/o SD3 Guidance)	34.88	30.67
SplatFlow (full model)	**34.85**	**31.43**

Table 7. **Impact of Stable Diffusion 3 Guidance.** The table compares the FID-10K and CLIPScore metrics for SplatFlow with and without SD3 guidance.

C.2. Ablation on Sampling Process

We modified the sampling process to enhance the quality of joint image, depth, and camera pose generation. Here, we present the ablation study for evaluating the impact of the main modifications: 1) early stopping of camera pose updates and 2) Stable Diffusion 3 guidance.

Effect of Early Stopping To assess the impact of early stopping for camera pose updates, we varied the stopping step (t_{stop}) at different values: 150 (the original, base setup), 100, 50, and 0 (no early stopping) during 200 total sampling steps. As shown in Table 6, stopping early at $t_{\text{stop}} = 150$ results in the best FID-10K and CLIPScore, indicating that fixing the camera poses early stabilizes the generated views. At $t_{\text{stop}} = 100$, there is a slight degradation in FID-10K and CLIPScore compared to $t_{\text{stop}} = 150$, and the performance further drops at $t_{\text{stop}} = 50$, which suggests that extending the camera pose updates introduces more degradation in the generated views. When no early stopping is applied ($t_{\text{stop}} = 0$), both metrics degrade significantly, highlighting the increased instability in camera pose updates over the entire sampling process. These results underline that stopping the ray updates early, preferably at $t_{\text{stop}} = 150$, is crucial for maintaining high-quality generation.

Effect of Stable Diffusion 3 guidance To measure the effects of Stable Diffusion 3 (SD3) guidance, we compared the original SplatFlow model, which includes SD3 guidance, against a variant without it. As shown in Table 7, removing SD3 guidance leads to a slight increase in FID-10K (34.88 compared to 34.85) and a notable drop in CLIPScore (from 31.43 to 30.67). These results indicate that SD3 guidance contributes positively to the consistency between generated images and the provided text prompts, thereby improving the alignment and quality of the generated outputs. Including SD3 guidance ensures better generalizability and alignment, particularly for smaller in-the-wild datasets.

Method	BRISQUE↓	NIQE↓	CLIPScore↑
DreamFusion [80]	90.2	10.48	-
Magic3D [52]	92.8	11.20	-
LatentNeRF [63]	88.6	9.19	-
SJC [103]	82.0	10.15	-
Fantasia3D [10]	69.6	7.65	-
ProlificDreamer [107]	61.5	7.07	-
Director3D [49]	37.1	6.41	32.0
Director3D (w/ SDS++) [49]	32.3	4.35	32.9
SplatFlow	**16.8**	5.88	28.9
SplatFlow (w/ SDS++)	19.6	**4.24**	**33.2**

Table 8. **Quantitative results in Single-Object-with-Surrounding set of T3Bench [26].** For the CLIPScore, we report our reproduced score due to an error in the measurement of Director3D [49].

C.3. More Results on Text-to-3DGS Generation

Generalizability evaluation Although our SplatFlow is trained on the MVImgNet [117] and the DL3DV [54] datasets, we conducted an experiment on Single-Ojbect-with-Surrounding sets of T3Bench [26] to validate the generalizability of our SplatFlow for unseen domain texts. Following evaluation protocols in Director3D [49], we utilized the BRISQUE [65] and the NIQE [66] to evaluate the image quality and the CLIPScore [28] to measure alignment with text prompts.

Table 8 demonstrates that our SplatFlow outperforms the previous text-to-3D generation methods across all metrics. Notably, our SplatFlow achieves a significantly lower score than other methods in the BRISQUE metric even without the refining process. Our SplatFlow performs worse than Director3D [49] in CLIPScore when both models are evaluated without a refining process, suggesting that SplatFlow has lower generalizability to text prompts due to its smaller training dataset. However, SplatFlow generates significantly higher-quality images compared to Director3D. This allows the refining process to focus primarily on aligning with the text prompts rather than enhancing image quality, leading to a substantial improvement in CLIPScore and ultimately surpassing Director3D [49].

Qualitative results As shown in Fig. 12, our SplatFlow generates realistic 3DGS and camera poses from various text prompts across MVImgNet [117], DL3DV [54], and T3Bench [26]. Interestingly, our SplatFlow primarily produces a straight-line camera trajectory for scenery-based descriptions, while creating a circular camera trajectory for object-centric descriptions.

C.4. More Results on 3DGS Editing

3D object replacement Our SplatFlow enables 3DGS editing through a modified SDEdit [62] combined with the inpainting process outlined in Algorithm 2. Specifically, for 3D object replacement, we perform an inversion on the masked region at $t = 190$ out of 200 total sampling steps, treating the
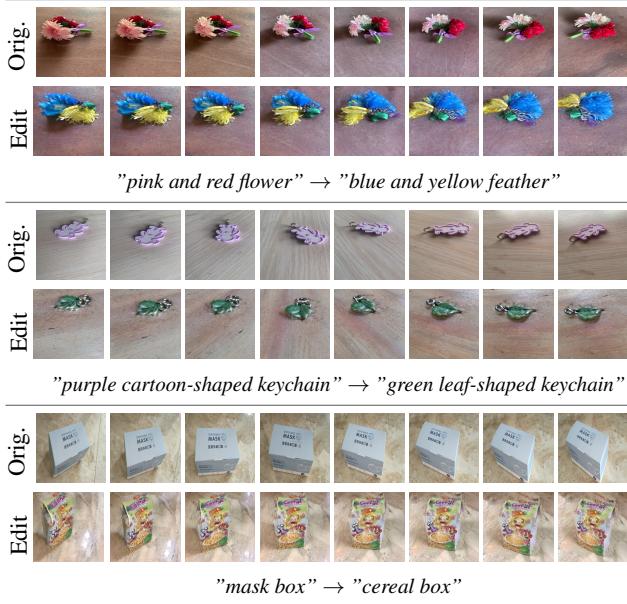
"pink and red flower" → "blue and yellow feather"

"purple cartoon-shaped keychain" → "green leaf-shaped keychain"

"mask box" → "cereal box"

Figure 8. **Additional qualitative results in 3DGS editing.**

"... sneakers ..." → "... sneakers *with a star mark* ..."

"... a *pink* ceramic vase ..." → "... a *blue* ceramic vase ..."

Figure 9. **Qualitative results on 3DGS editing with user-provided strokes.** Despite the rough strokes applied to the rendered scene, our SplatFlow enables seamless and natural 3DGS editing.

Figure 10. **Depth map visualization.** Given multi-view images, we show generated depths that are jointly inpainted with camera poses.

remaining area as known regions by utilizing the foreground mask of the main object. Additional qualitative results on the 3D object replacement are presented in Fig. 8, confirming its effectiveness in 3DGS editing on various objects.

3DGS editing with strokes Interestingly, our SplatFlow can selectively edit specific portions of the generated 3DGS based on user-provided input strokes. Specifically, we perform an inversion on all rendered multi-view images with strokes at $t = 100$ out of 200 total sampling steps, followed by denoising the latents using edited captions. As shown in Fig. 9, even with rough stroke inputs that are 3D inconsistent, the edited 3DGS maintains a highly natural appearance.

D. Anaylsis and Discussion

D.1. Depth Map Visualization

Note that better performance is achieved when the depth map from DepthAnythingV2 [112] is not used for camera pose estimation. Figure 10 shows the generated depth maps obtained by jointly generating depth and ray latents from multi-view images. Notably, the generated depth maps capture the details of the given images more effectively than the ground truth provided by DepthAnythingV2 [112]. Therefore, more detailed depth maps allow our multi-view RF to achieve more accurate camera pose estimation.

D.2. 3D Consistency Analysis for the GSDecoder

In this section, we explore the role of the GSDecoder by comparing the multi-view generated images with the 3D Gaussian Splatting (3DGS) rendered outputs, as shown in Fig. 11. The multi-view images are decoded using the Stable Diffusion 3 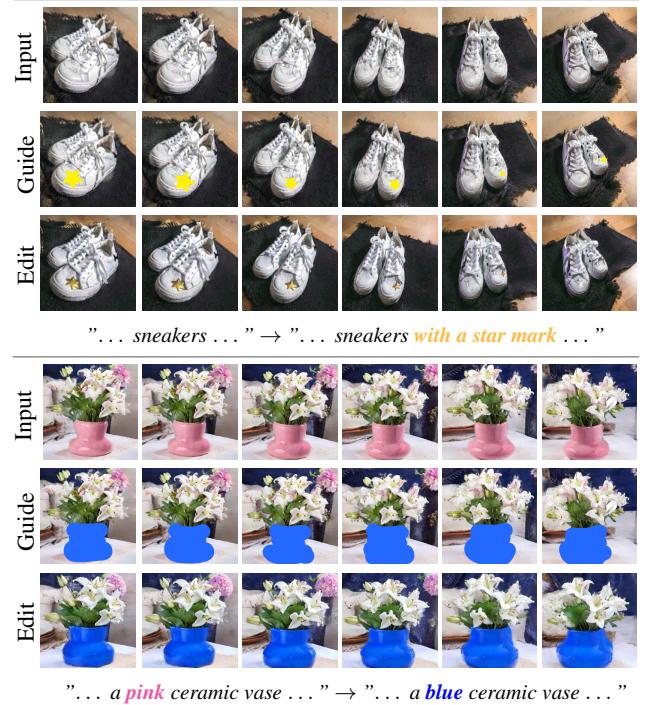decoder, while the 3DGS is rendered using our GSDecoder in a feed-forward manner. Our objective is to analyze how the GSDecoder contributes to achieving consistency across the 3D space by transforming the multi-view latent representations into a coherent 3D representation.

From our analysis, we observe that the GSDecoder can help mitigate inconsistencies present in the multi-view generated images. Specifically, there are instances where slight variations in the appearance of objects, such as colors or shapes, are noticeable in the multi-view generated images, which can lead to a lack of 3D coherence in the generated scene. The GSDecoder processes these inconsistencies by smoothing them, resulting in more consistent 3DGS parameters across multiple views. In the case of the red apple on a wooden surface, the GSDecoder addresses inconsistencies in
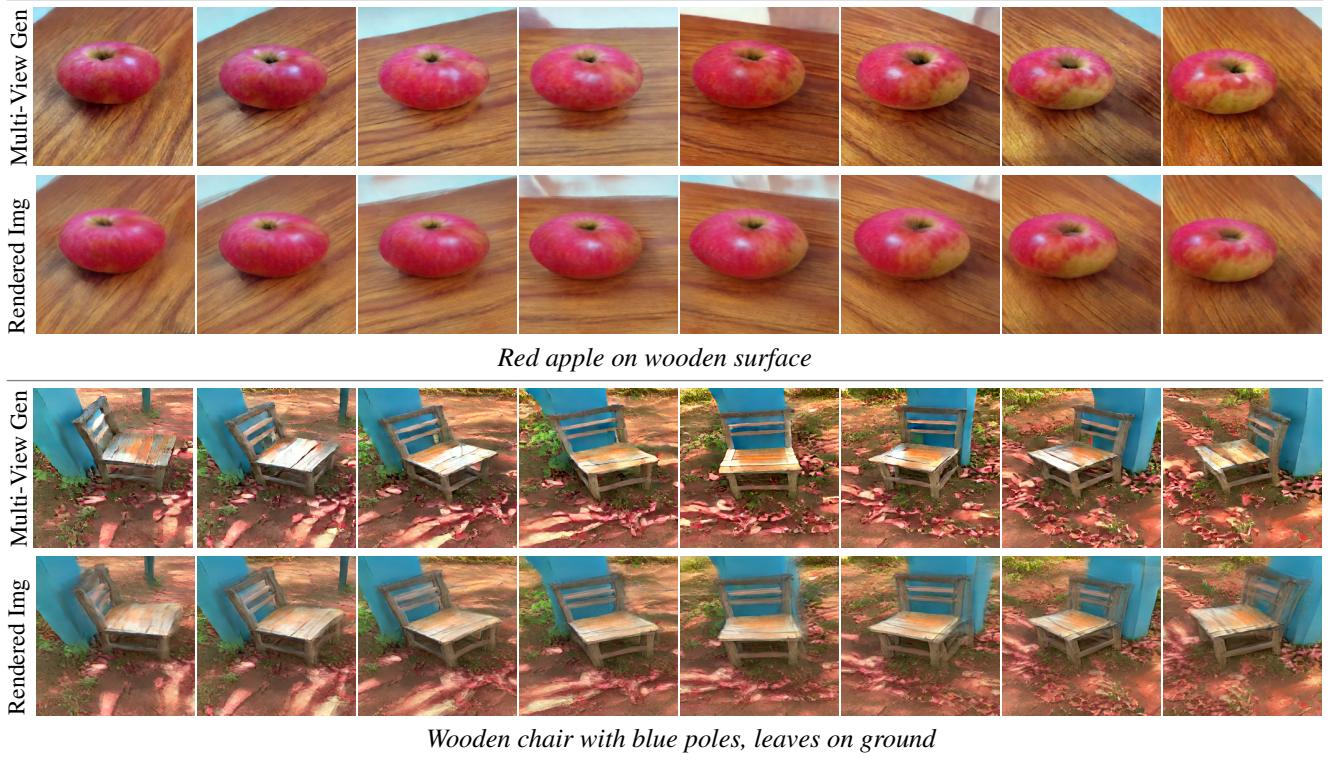
Figure 11. **GSDecoder analysis.** Comparison between multi-view generated images (top row of each pair) and 3D Gaussian Splatting (3DGS) rendered outputs (bottom row of each pair) using the GSDecoder. The GSDecoder helps smooth out inconsistencies present in the multi-view generated images, such as variations in color and shape, resulting in more coherent 3D representations. For example, inconsistencies in the apple's color and shape, and in the seat cushion's color of the wooden chair, are corrected.

the apple's color and shape that are evident in the multi-view images. For the wooden chair with blue poles and leaves on the ground, the GSDecoder helps align the color of the seat cushion, making it more consistent across different views. Overall, the GSDecoder prioritizes 3D consistency at the cost of slightly blurring some details, ultimately enhancing the coherence of the generated scene.
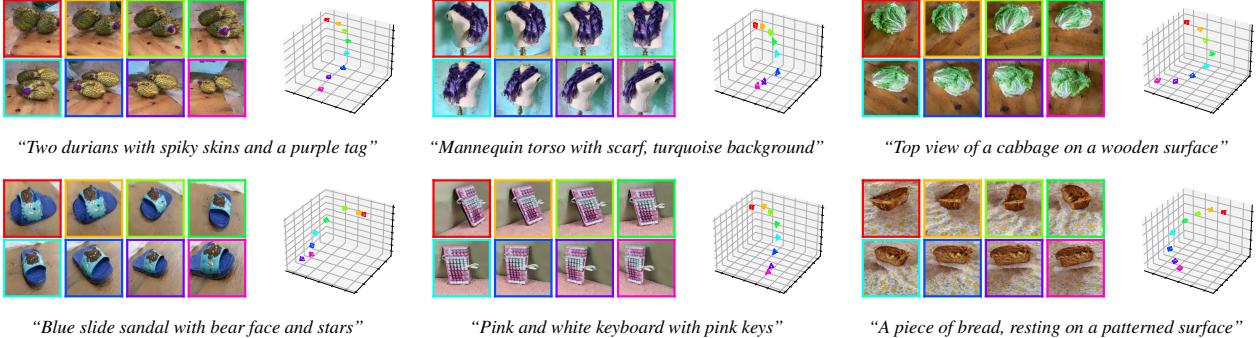
D.3. Discussions

Inference time   It takes about 20 seconds to generate one 3D scene from text prompts and 5 minutes for the refining process (*i.e.*, SDS++ [49]). This is similar to Director3D [49] which also utilizes the same refining process.

Future works and limitations   As our SplatFlow is trained only on MVImgNet [117] and DL3DV [54], its generalizability to unseen text prompts has room for improvement. Specifically, training with a large text-to-image dataset [89] and other multi-view image datasets [82] can improve the generalizability. Additionally, we believe that the inversion technique or inpainting method suitable for the rectified flow model can be combined with our multi-view RF model to achieve better quality 3DGS editing.

Ethical Considerations   Advancements in 3D generation technology raise several ethical considerations that require

careful attention. A significant concern is the potential misuse of generated 3D content, as it could be exploited to create deceptive or misleading visuals. Fabricated content could be presented as authentic, potentially leading to harm or misinformation. To prevent the misuse of this technology, it is crucial to establish clear guidelines for responsible use and enforce ethical standards. Implementing robust safeguards and obtaining informed consent, especially when processing images that contain personal information, are crucial steps to prevent the misuse of this technology.

20

MVImgNet [117] validation set

"Two durians with spiky skins and a purple tag"

"Mannequin torso with scarf, turquoise background"

"Top view of a cabbage on a wooden surface"

"Blue slide sandal with bear face and stars"

"Pink and white keyboard with pink keys"

"A piece of bread, resting on a patterned surface"

DL3DV [54] validation set

"A stone building with a tiled floor and an arched window"

"A modern office space with a bench, chairs, and tables"

"A serene park with picnic tables, benches, and a gazebo"

"A blue cloth on a wooden surface with three peaches"

"Rows of colorful books on shelves in a bookstore"

"A modern building with a unique facade and large windows"

T3Bench [26] - Single-Object-with-Surrounding set

"A stack of pancakes on a breakfast table"

"A jar of homemade jam on a kitchen counter"

"A bright sunflower in a field"

"A colorful parrot on a jungle tree"

"A pair of hiking boots on a trail"

"A rainbow over a waterfall"

"A snowman wearing a scarf in a winter landscape"

"A striped beach umbrella standing tall on a sandy beach"

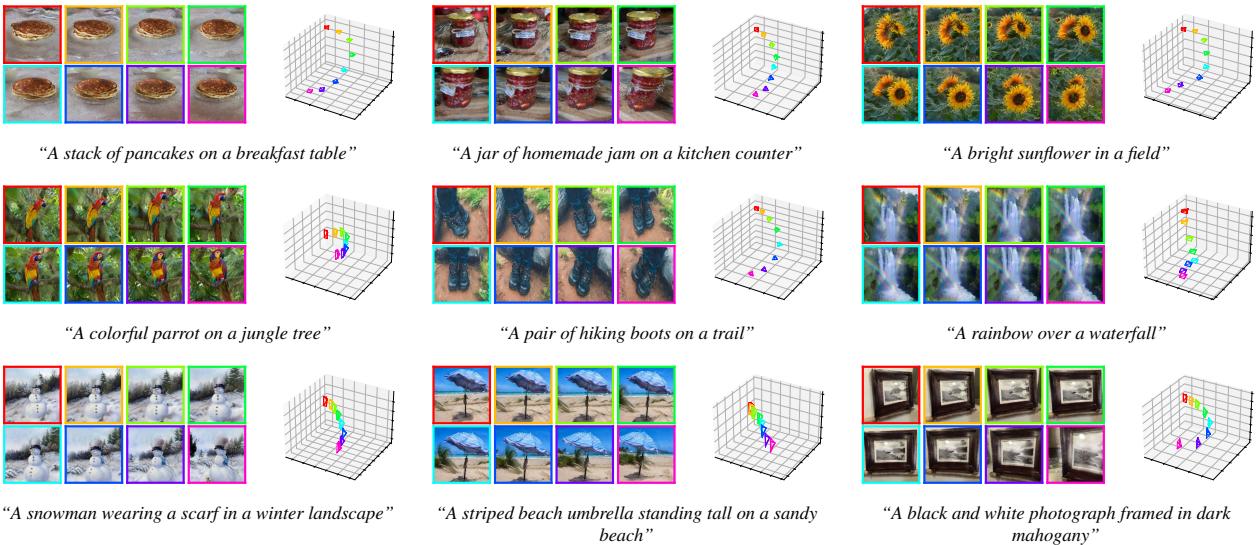*"A black and white photograph framed in dark mahogany"*

Figure 12. **Additional qualitative results in 3DGS generation on MVImgNet [117], DL3DV [54], and T3Bench [26].** We show eight rendered scenes and camera poses from given text prompts, where image border colors match each camera.