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

Supplementary Material

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 [7], 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 [45], and vision-aided GAN loss [15]. Specifically, the vision-aided GAN loss leverages backbones from DINO [4] and CLIP [31], and we incorporated differentiable augmentation [13] along with a multi-level version of hinge loss[1]. Therefore, our loss can be represented as:

$$L_{\text{decoder}} = w_1 L_{\text{mse}} + w_2 L_{\text{LPIPS}} + w_3 L_{\text{vision-aided}}, \quad (1)$$

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 [33]. Specifically, w_3 is determined 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 [7][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 [7], we applied an SNR sampler and used three text encoders.

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 [44], 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 [7] 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 [21, 43]. Applying this guidance early ensures consistency between

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

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

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:

for $i = N, \ldots, 1$ **do**
  $\hat{\boldsymbol{v}}_{t_i} \leftarrow \boldsymbol{u}_\theta(\boldsymbol{Y}_{t_i}, t_i)$
  if $i \geq t_{stop}$ **then**
    if $N - i \equiv 0 \mod 3 \ \ \& \ \ i \neq t_{stop}$ **then**
      $\hat{\boldsymbol{v}}_{t_i}[:n] \leftarrow \boldsymbol{v}_\phi(\boldsymbol{Y}_{t_i}[:n], t_i)$ ▷ *Replace to SD3*
    end if
    $\tilde{\boldsymbol{Y}}_{t_0} \leftarrow \boldsymbol{Y}_{t_i} - t_i \boldsymbol{u}_\theta(\boldsymbol{Y}_{t_i}, t_i)$ ▷ *Predict Destination*
    $\langle \boldsymbol{d}^{(1:K)}, \boldsymbol{m}^{(1:K)} \rangle \leftarrow \tilde{\boldsymbol{Y}}_{t_0}[2n:]$ ▷ *Extract Ray Latent*
    for $j = 1, \ldots, K$ **do**
      $\langle \boldsymbol{K}^j, \boldsymbol{R}^j, \boldsymbol{T}^j \rangle \leftarrow$ ray_optimize$(\langle \boldsymbol{d}^j, \boldsymbol{m}^j \rangle)$
    end for
    $\langle \boldsymbol{K}, \boldsymbol{R}^{(1:K)}, \boldsymbol{T}^{(1:K)} \rangle \leftarrow$ shared_K$(\langle \boldsymbol{K}, \boldsymbol{R}, \boldsymbol{T} \rangle^{(1:K)})$
    $\langle \boldsymbol{d}^{(1:K)}, \boldsymbol{m}^{(1:K)} \rangle \leftarrow$ plücker$(\langle \boldsymbol{K}, \boldsymbol{R}^{(1:K)}, \boldsymbol{T}^{(1:K)} \rangle)$
    $\boldsymbol{r}_{t_0} \leftarrow \langle \boldsymbol{d}^{(1:K)}, \boldsymbol{m}^{(1:K)} \rangle$ ▷ *Update Ray Destination*
  end if
  $\boldsymbol{Y}_{t_{i-1}} \leftarrow \boldsymbol{Y}_{t_i} + (t_{i-1} - t_i)\hat{\boldsymbol{v}}_{t_i}(\boldsymbol{Y}_{t_i}, t_i)$
  $\boldsymbol{z} \sim \mathcal{N}(0, I)$
  $\boldsymbol{Y}_{t_{i-1}}[2n:] \leftarrow (1 - t_{i-1})\boldsymbol{r}_{t_0} + t_{i-1}\boldsymbol{z}$
end for
return $\boldsymbol{Y}_{t_0}, \boldsymbol{K}, \boldsymbol{R}^{(1:K)}, \boldsymbol{T}^{(1:K)}$

multi-view images and depths while improving their quality. Also, we use the dual-mode toggling approach similar to Dual3D [16], 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 [44] as:

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

$$\boldsymbol{P}^j = \underset{\|\boldsymbol{H}\|=1}{\arg\min} \sum_{i=1}^{h \times w} \|\boldsymbol{H}\boldsymbol{d}_i^j \times \boldsymbol{u}_i\|, \tag{3}$$

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 [14] optimizer with 10 iterations (which adds negligible overhead), ensuring that all views share the same intrinsic matrix as:

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:

for $i = N, \ldots, 1$ **do**
  if $i \geq t_{stop}$ **then**
    $\tilde{\boldsymbol{Y}}_{t_0} \leftarrow \boldsymbol{Y}_{t_i} - t_i \boldsymbol{u}_\theta(\boldsymbol{Y}_{t_i}, t_i)$ ▷ *Predict Destination*
    $\langle \boldsymbol{d}^{(1:K)}, \boldsymbol{m}^{(1:K)} \rangle \leftarrow \tilde{\boldsymbol{Y}}_{t_0}[2n:]$ ▷ *Extract Ray Latent*
    for $j = 1, \ldots, K$ **do**
      $\langle \boldsymbol{K}^j, \boldsymbol{R}^j, \boldsymbol{T}^j \rangle \leftarrow$ ray_optimize$(\langle \boldsymbol{d}^j, \boldsymbol{m}^j \rangle)$
    end for
    $\langle \boldsymbol{K}, \boldsymbol{R}^{(1:K)}, \boldsymbol{T}^{(1:K)} \rangle \leftarrow$ shared_K$(\langle \boldsymbol{K}, \boldsymbol{R}, \boldsymbol{T} \rangle^{(1:K)})$
    $\langle \boldsymbol{d}^{(1:K)}, \boldsymbol{m}^{(1:K)} \rangle \leftarrow$ plücker$(\langle \boldsymbol{K}, \boldsymbol{R}^{(1:K)}, \boldsymbol{T}^{(1:K)} \rangle)$
    $\boldsymbol{r}_{t_0} \leftarrow \langle \boldsymbol{d}^{(1:K)}, \boldsymbol{m}^{(1:K)} \rangle$ ▷ *Update Ray Destination*
  end if
  $\boldsymbol{Y}_{t_{i-1}}^{\text{unknown}} \leftarrow \boldsymbol{Y}_{t_i} + (t_{i-1} - t_i)\boldsymbol{u}_\theta(\boldsymbol{Y}_{t_i}, t_i)(\boldsymbol{Y}_{t_i}, t_i)$
  $\boldsymbol{z} \sim \mathcal{N}(0, I)$
  $\boldsymbol{Y}_{t_{i-1}}^{\text{unknown}}[2n:] \leftarrow (1 - t_{i-1})\boldsymbol{r}_{t_0} + t_{i-1}\boldsymbol{z}$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$
  $Y_{t_{i-1}}^{\text{known}} = (1 - t_{i-1})Y_{t_0}^{\text{known}} + t_{i-1}\boldsymbol{\epsilon}$
  $Y_{t_{i-1}} = m \odot Y_{t_{i-1}}^{\text{known}} + (1 - m) \odot Y_{t_{i-1}}^{\text{unknown}}$
end for
return $\boldsymbol{Y}_{t_0}, \boldsymbol{K}, \boldsymbol{R}^{(1:K)}, \boldsymbol{T}^{(1:K)}$

$$\langle \boldsymbol{K}, \boldsymbol{R}^{(1:K)} \rangle = \underset{\|\boldsymbol{R}\|=1}{\arg\min} \sum_{j=1}^{K} \sum_{i=1}^{h \times w} \|\boldsymbol{R}^j \boldsymbol{d}_i^j \times \boldsymbol{u}_{\boldsymbol{K},i}\|, \tag{4}$$

where $\boldsymbol{u}_{\boldsymbol{K}}$ 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 employ different classifier-free guidance [12] 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 [23] 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. Fur-

thermore, we utilize RePaint [23] 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 [43], 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 [21] 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 SplatFlow. 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 [22] 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 [41][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 [29][4] to assess the distance between the generated and reference images. 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."
- Target Caption: "A sleek metallic spoon with a
    reflective surface on a plaid fabric."
```

[3] https://huggingface.co/depth-anything/Depth-Anything-V2-Small

[4] Implementation available at https://github.com/GaParmar/clean-fid

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 1. **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.

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 [5] and MVInpainter [3] as baselines by using their official implementations with the following configurations:

- **DGE** [5]: 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** [3]: 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 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 [45], SSIM [39], and FID [11]. For FID calculations, we sampled 50K reference images.

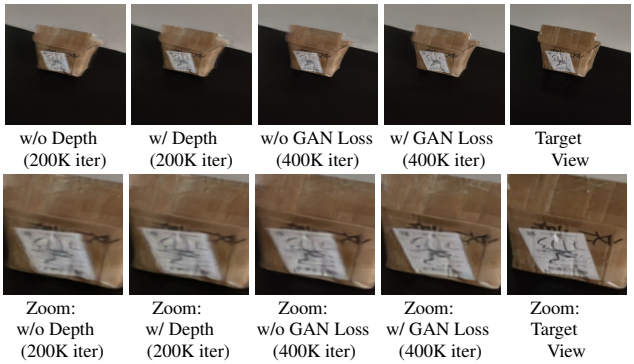

Figure 1. **Ablation study on GSDecoder design choices.** The first row shows the original views, while the second row provides zoomed-in details for better visualization. Incorporating depth latents and vision-aided GAN loss enhances the realism and quality of generated 3D Gaussian Splatting (3DGS) scenes.

The results of this ablation study are illustrated in Table 1. 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 GSDecoder variants is presented in Fig. 1. 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.

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

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

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 2, 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 3, 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.

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

Generalizability evaluation Although our SplatFlow is trained on the MVImgNet [43] and the DL3DV [21] datasets, we conducted an experiment on Single-Ojbect-with-Surrounding sets of T3Bench [9] to validate the generalizability of our SplatFlow for unseen domain texts. Following evaluation protocols in Director3D [17], we utilized the BRISQUE [26] and the NIQE [27] to evaluate the image quality and the CLIPScore [10] to measure alignment with text prompts.

Method	BRISQUE↓	NIQE↓	CLIPScore↑
DreamFusion [30]	90.2	10.48	-
Magic3D [20]	92.8	11.20	-
LatentNeRF [25]	88.6	9.19	-
SJC [38]	82.0	10.15	-
Fantasia3D [6]	69.6	7.65	-
ProlificDreamer [40]	61.5	7.07	-
Director3D [17]	37.1	6.41	32.0
Director3D (w/ SDS++) [17]	32.3	4.35	32.9
SplatFlow	**16.8**	5.88	28.9
SplatFlow (w/ SDS++)	19.6	**4.24**	**33.2**

Table 4. **Quantitative results in Single-Object-with-Surrounding set of T3Bench [9].** For the CLIPScore, we report our reproduced score due to an error in the measurement of Director3D [17].

Table 4 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 [17] 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 [17].

Qualitative results As shown in Fig. 6, our SplatFlow generates realistic 3DGS and camera poses from various text prompts across MVImgNet [43], DL3DV [21], and T3Bench [9]. 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 [24] 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 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. 2, confirming

Figure 2. **Additional qualitative results in 3DGS editing.**

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. 3, 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 [41] is not used for camera pose estimation. Figure 4 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 [41]. 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. 5. 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

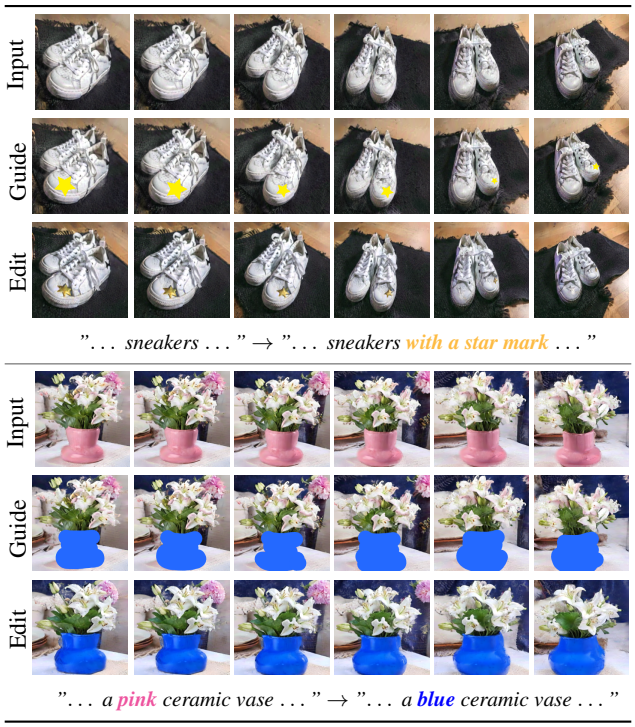

Figure 3. **Qualitative results on 3DGS editing with user-provided strokes.** Despite the rough strokes applied to the rendered scene, our SplatFlow enables seamless and natural 3DGS editing.

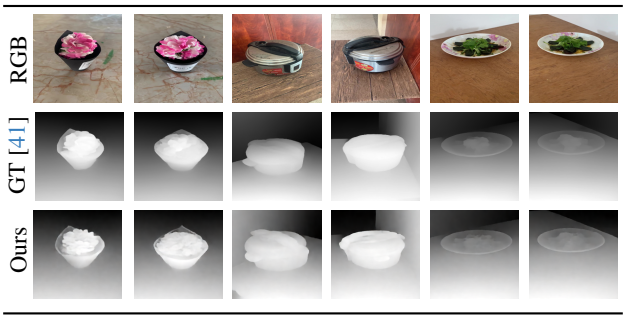

Figure 4. **Depth map visualization.** Given multi-view images, we show generated depths that are jointly inpainted with camera poses.

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 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

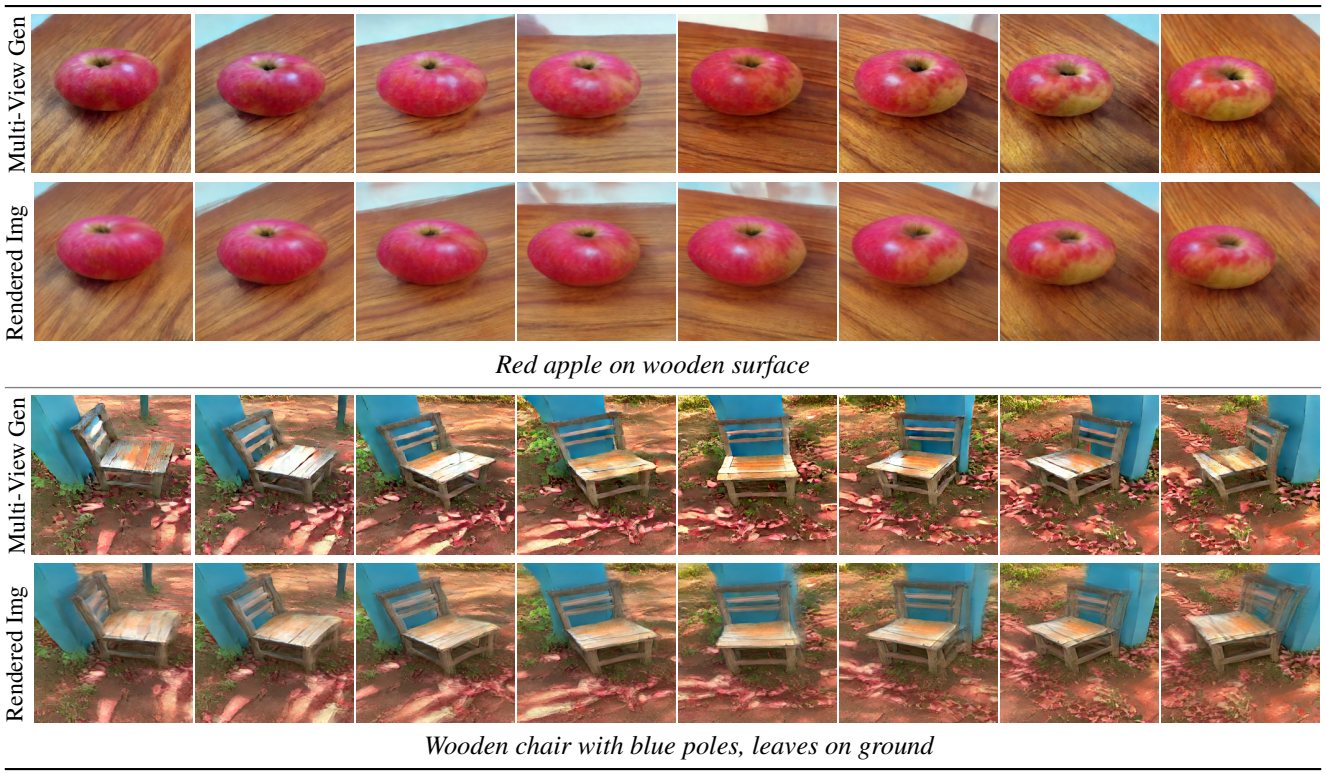

Figure 5. **GSDecoder analysis.** Comparison between multi-view generated images (top row of each pair) and 3D Gaussian Splatting (3DGS) rendered outputs (bottom row of each pair) using the GSDecoder. The GSDecoder helps smooth out inconsistencies present in the multi-view generated images, such as variations 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.

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++ [17]). This is similar to Director3D [17] which also utilizes the same refining process.

Future works and limitations As our SplatFlow is trained only on MVImgNet [43] and DL3DV [21], its generalizability to unseen text prompts has room for improvement. Specifically, training with a large text-to-image dataset [34] and other multi-view image datasets [32] 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.

D.4. Extended Related Works

We sincerely appreciate the reviewers' valuable feedback and have decided to add relevant discussions in this section. Previous works have attempted to ensure 3D consistency by leveraging diffusion models [2, 36]. Concurrent works have further taken a step toward direct 3D Gaussian Splatting (3DGS) by replacing conventional diffusion decoders with Gaussian Splatting-based decoders [8, 18, 19, 28, 35, 37, 42]. We believe this direction is not only more versatile but also paves the way for more straightforward and effective editing through generative modeling.

Figure 6. **Additional qualitative results in 3DGS generation on MVImgNet [43], DL3DV [21], and T3Bench [9].** We show eight rendered scenes and camera poses from given text prompts, where image border colors match each camera.

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. 2

[2] Titas Anciukevičius, Fabian Manhardt, Federico Tombari, and Paul Henderson. Denoising diffusion via image-based rendering. In *The Twelfth International Conference on Learning*

Representations, 2024. 7

[3] 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. 4

[4] 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. 1

[5] Minghao Chen, Iro Laina, and Andrea Vedaldi. Dge: Direct gaussian 3d editing by consistent multi-view editing. *arXiv preprint arXiv:2404.18929*, 2024. 4

[6] 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. 5

[7] 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. 1

[8] Hyojun Go, Byeongjun Park, Hyelin Nam, Byung-Hoon Kim, Hyungjin Chung, and Changick Kim. Videorfsplat: Direct scene-level text-to-3d gaussian splatting generation with flexible pose and multi-view joint modeling, 2025. 7

[9] Yuze He, Yushi Bai, Matthieu Lin, Wang Zhao, Yubin Hu, Jenny Sheng, Ran Yi, Juanzi Li, and Yong-Jin Liu. T^3bench: Benchmarking current progress in text-to-3d generation, 2023. 5, 8

[10] 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. 5

[11] 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. 4

[12] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 2

[13] 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. 1

[14] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[15] 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. 1

[16] 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. 2

[17] 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. 5, 7

[18] Hanwen Liang, Junli Cao, Vidit Goel, Guocheng Qian, Sergei Korolev, Demetri Terzopoulos, Konstantinos N Plataniotis, Sergey Tulyakov, and Jian Ren. Wonderland: Navigating 3d scenes from a single image. *arXiv preprint arXiv:2412.12091*, 2024. 7

[19] Chenguo Lin, Panwang Pan, Bangbang Yang, Zeming Li, and Yadong Mu. Diffsplat: Repurposing image diffusion models for scalable gaussian splat generation. *arXiv preprint arXiv:2501.16764*, 2025. 7

[20] 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. 5

[21] 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. 1, 3, 5, 7, 8

[22] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 3

[23] 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, 3

[24] 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. 5

[25] 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. 5

[26] 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. 5

[27] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal processing letters*, 20(3):209–212, 2012. 5

[28] Byeongjun Park, Hyojun Go, Hyelin Nam, Byung-Hoon Kim, Hyungjin Chung, and Changick Kim. Steerx: Creating any camera-free 3d and 4d scenes with geometric steering. *arXiv preprint arXiv:2503.12024*, 2025. 7

[29] 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. 3

[30] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 5

[31] 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. 1

[32] 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. 7

[33] 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. 1

[34] 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. 7

[35] Katja Schwarz, Norman Mueller, and Peter Kontschieder. Generative gaussian splatting: Generating 3d scenes with video diffusion priors. *arXiv preprint arXiv:2503.13272*, 2025. 7

[36] Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Viewset diffusion:(0-) image-conditioned 3d generative models from 2d data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8863–8873, 2023. 7

[37] Stanislaw Szymanowicz, Jason Y Zhang, Pratul Srinivasan, Ruiqi Gao, Arthur Brussee, Aleksander Holynski, Ricardo Martin-Brualla, Jonathan T Barron, and Philipp Henzler. Bolt3d: Generating 3d scenes in seconds. *arXiv preprint arXiv:2503.14445*, 2025. 7

[38] 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. 5

[39] 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. 4

[40] 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. 5

[41] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv preprint arXiv:2406.09414*, 2024. 3, 6

[42] Yuanbo Yang, Jiahao Shao, Xinyang Li, Yujun Shen, Andreas Geiger, and Yiyi Liao. Prometheus: 3d-aware latent diffusion models for feed-forward text-to-3d scene generation. *arXiv preprint arXiv:2412.21117*, 2024. 7

[43] 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. 1, 3, 5, 7, 8

[44] 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. 1, 2

[45] 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. 1, 4