ViewDiff: 3D-Consistent Image Generation with Text-to-Image Models

Lukas Höllein[1,2] Aljaž Božič[2] Norman Müller[2] David Novotny[2] Hung-Yu Tseng[2]
Christian Richardt[2] Michael Zollhöfer[2] Matthias Nießner[1]

[1]Technical University of Munich [2]Meta

https://lukashoel.github.io/ViewDiff/

Figure 1. **Multi-view consistent image generation.** Our method takes as input a text description, or any number of posed input images, and generates high-quality, multi-view consistent images of a real-world 3D object in authentic surroundings from any desired camera poses.

Abstract

3D asset generation is getting massive amounts of attention, inspired by the recent success of text-guided 2D content creation. Existing text-to-3D methods use pretrained text-to-image diffusion models in an optimization problem or fine-tune them on synthetic data, which often results in non-photorealistic 3D objects without backgrounds. In this paper, we present a method that leverages pretrained text-to-image models as a prior, and learn to generate multi-view images in a single denoising process from real-world data. Concretely, we propose to integrate 3D volume-rendering and cross-frame-attention layers into each block of the existing U-Net network of the text-to-image model. Moreover, we design an autoregressive generation that renders more 3D-consistent images at any viewpoint. We train our model on real-world datasets of objects and showcase its capabilities to generate instances with a variety of high-quality shapes and textures in authentic surroundings. Compared to the existing methods, the results generated by our method are consistent, and have favorable visual quality (-30% FID, -37% KID).

1. Introduction

In recent years, text-to-image (T2I) diffusion models [31, 35] have emerged as cutting-edge technologies, revolutionizing high-quality and imaginative 2D content creation guided by text descriptions. These frameworks have found widespread applications, including extensions such as ControlNet [59] and DreamBooth [34], showcasing their versatility and potential. An intriguing direction in this domain is to use T2I models as powerful 2D priors for generating three-dimensional (3D) assets. How can we effectively use these models to create photo-realistic and diverse 3D assets?

Existing methods like DreamFusion [29], Fantasia3D [5], and ProlificDreamer [52] have demonstrated exciting results by optimizing a 3D representation through score distillation sampling [29] from pretrained T2I diffusion models. The

3D assets generated by these methods exhibit compelling diversity. However, their visual quality is not consistently as high as that of the images generated by T2I models. A key step to obtaining 3D assets is the ability to generate consistent multi-view images of the desired objects and their surroundings. These images can then be fitted to 3D representations like NeRF [26] or NeuS [50]. HoloDiffusion [19] and ViewsetDiffusion [42] train a diffusion model *from scratch* using multi-view images and output 3D-consistent images. GeNVS [4] and DFM [46] additionally produce object surroundings, thereby increasing the realism of the generation. These methods ensure (photo)-realistic results by training on real-world 3D datasets [33, 62]. However, these datasets are orders of magnitude smaller than the 2D dataset used to train T2I diffusion models. As a result, these approaches produce realistic but less-diverse 3D assets. Alternatively, recent works like Zero-1-to-3 [24] and One-2-3-45 [23] leverage a pretrained T2I model and fine-tune it for 3D consistency. These methods successfully preserve the diversity of generated results by training on a large synthetic 3D dataset [7]. Nonetheless, the produced objects can be less photo-realistic and are without surroundings.

In this paper, we propose a method that leverages the 2D priors of the pretrained T2I diffusion models to produce *photo-realistic* and *3D-consistent* 3D asset renderings. As shown in the first two rows of Fig. 1, the input is a text description or an image of an object, along with the camera poses of the desired rendered images. The proposed approach produces multiple images of the same object in a single forward pass. Moreover, we design an autoregressive generation scheme that allows to render more images at *any* novel viewpoint (Fig. 1, third row). Concretely, we introduce projection and cross-frame-attention layers, that are strategically placed into the existing U-Net architecture, to encode explicit 3D knowledge about the generated object (see Fig. 2). By doing so, our approach paves the way to fine-tune T2I models on real-world 3D datasets, such as CO3D [33], while benefiting from the large 2D prior encoded in the pretrained weights. Our generated images are consistent, diverse, and realistic renderings of objects.

To summarize, our contributions are:

- a method that utilizes the pretrained 2D prior of text-to-image models and turns them into 3D-consistent image generators. We train our approach on real-world multi-view datasets, allowing us to produce realistic and high-quality images of objects and their surroundings (Sec. 3.1).
- a novel U-Net architecture that combines commonly used 2D layers with 3D-aware layers. Our projection and cross-frame-attention layers encode explicit 3D knowledge into each block of the U-Net architecture (Sec. 3.2).
- an autoregressive generation scheme that renders images of a 3D object from *any* desired viewpoint *directly* with our model in a 3D-consistent way (Sec. 3.3).

2. Related Work

Text-To-2D. Denoising diffusion probabilistic models (DDPM) [14] model a data distribution by learning to invert a Gaussian noising process with a deep network. Recently, DDPMs were shown to be superior to generative adversarial networks [8], becoming the state-of-the-art framework for image generation. Soon after, large text-conditioned models trained on billion-scale data were proposed in Imagen [35] or Dall-E 2 [31]. While [8] achieved conditional generation via guidance with a classifier, [13] proposed classifier-free guidance. ControlNet [59] proposed a way to tune the diffusion outputs by conditioning on various modalities, such as image segmentation or normal maps. Similar to ControlNet, our method builds on the strong 2D prior of a pretrained text-to-image (T2I) model. We further demonstrate how to adjust this prior to generate 3D-consistent images of objects.

Text-To-3D. 2D DDPMs were applied to the generation of 3D shapes [30, 37, 44, 47, 49, 55, 64] or scenes [10, 15, 45] from text descriptions. DreamFusion [29] proposed score distillation sampling (SDS) which optimizes a 3D shape whose renders match the belief of the DDPM. Improved sample quality was achieved by a second-stage mesh optimization [5, 22], and smoother SDS convergence [38, 52]. Several methods use 3D data to train a novel-view synthesis model whose multi-view samples can be later converted to 3D, e.g. conditioning a 2D DDPM on an image and a relative camera motion to generate novel views [24, 53]. However, due to no explicit modelling of geometry, the outputs are view-inconsistent. Consistency can be improved with epipolar attention [48, 63], or optimizing a 3D shape from multi-view proposals [23]. Our work fine-tunes a 2D T2I model to generate renders of a 3D object; however, we propose explicit 3D unprojection and rendering operators to improve view-consistency. Concurrently, SyncDreamer [25] also add 3D layers in their 2D DDPM. We differ by training on real data with backgrounds and by showing that autoregressive generation is sufficient to generate consistent images, making the second 3D reconstruction stage expendable.

Diffusion on 3D Representations. Several works model the distribution of 3D representations. While DiffRF [28] leverages ground-truth 3D shapes, HoloDiffusion [19] is supervised only with 2D images. HoloFusion [18] extends this work with a 2D diffusion render post-processor. Images can also be denoised by rendering a reconstructing 3D shape [1, 42]. Unfortunately, the limited scale of existing 3D datasets prevents these 3D diffusion models from extrapolating beyond the training distribution. Instead, we exploit a large 2D pretrained DDPM and add 3D components that are tuned on smaller-scale multi-view data. This leads to improved multi-view consistency while maintaining the expressivity brought by pretraining on billion-scale image data.

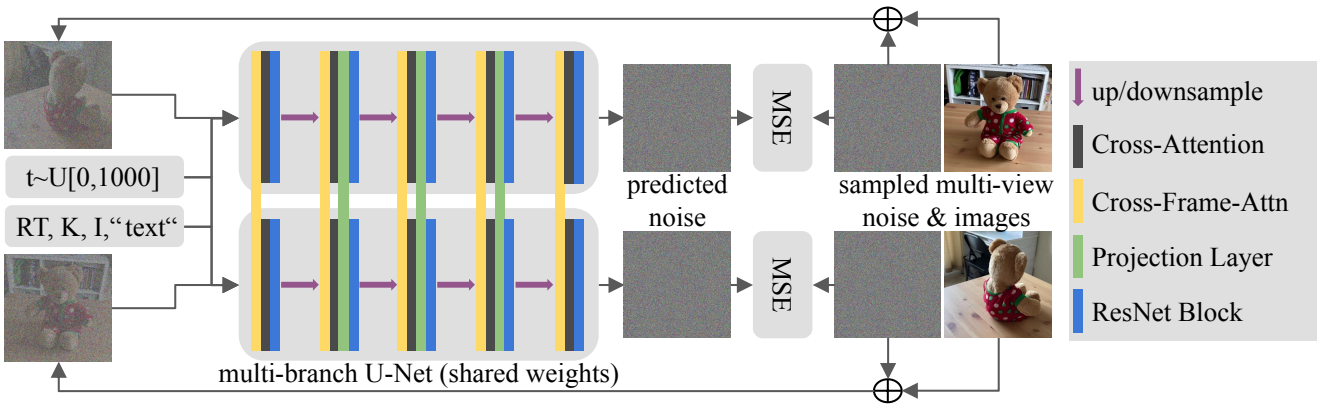


Figure 2. **Method Overview.** We augment the U-Net architecture of pretrained text-to-image models with new layers in every U-Net block. These layers facilitate communication between multi-view images in a batch, resulting in a denoising process that jointly produces 3D-consistent images. First, we replace self-attention with cross-frame-attention (yellow) which compares the spatial features of all views. We condition all attention layers on pose (RT), intrinsics (K), and intensity (I) of each image. Second, we add a projection layer (green) into the inner blocks of the U-Net. It creates a 3D representation from multi-view features and renders them into 3D-consistent features. We fine-tune the U-Net using the diffusion denoising objective (Eq. 3) at timestep t, supervised from captioned multi-view images.

3. Method

We propose a method that produces 3D-consistent images from a given text or posed image input (see Fig. 1 top/mid). Concretely, given desired output poses, we jointly generate all images corresponding to the condition. We leverage pretrained text-to-image (T2I) models [31, 35] and fine-tune them on multi-view data [33]. We propose to augment the existing U-Net architecture by adding new layers into each block (see Fig. 2). At test time, we can condition our method on multiple images (see Fig. 1 bottom), which allows us to autoregressively render the same object from *any* viewpoint *directly* with the diffusion model (see Sec. 3.3).

3.1. 3D-Consistent Diffusion

Diffusion models [14, 39] are a class of generative models that learn the probability distribution $p_\theta(x_0) = \int p_\theta(x_{0:T}) dx_{1:T}$ over data $x_0 \sim q(x_0)$ and latent variables $x_{1:T} = x_1, \ldots, x_T$. Our method is based on pretrained text-to-image models, which are diffusion models $p_\theta(x_0 \mid c)$ with an additional text condition c. For clarity, we drop the condition c for the remainder of this section.

To produce multiple images $x_0^{0:N}$ at once, which are 3D-consistent with each other, we seek to model their joint probability distribution $p_\theta(x_0^{0:N}) = \int p_\theta(x_{0:T}^{0:N}) dx_{1:T}^{0:N}$. Similarly to concurrent work by Liu et al. [25], we generate one set of images $p_\theta(x_0^{0:N})$ by adapting the *reverse process* of DDPMs [14] as a Markov chain over all images jointly:

$$p_\theta(x_{0:T}^{0:N}) := p(x_T^{0:N}) \prod_{t=1}^{T} \prod_{n=0}^{N} p_\theta(x_{t-1}^n \mid x_t^{0:N}), \quad (1)$$

where we start the generation from Gaussian noise sampled separately per image $p(x_T^n) = \mathcal{N}(x_T^n; \mathbf{0}, \mathbf{I})$, $\forall n \in [0, N]$. We gradually denoise samples $p_\theta(x_{t-1}^n \mid x_t^{0:N}) = \mathcal{N}(x_{t-1}; \mu_\theta^n(x_t^{0:N}, t), \sigma_t^2 \mathbf{I})$ by predicting the per-image mean $\mu_\theta^n(x_t^{0:N}, t)$ through a neural network μ_θ that is shared between all images. Importantly, at each step, the model uses the previous states $x_t^{0:N}$ of all images, i.e., there is communication between images during the model prediction. We refer to Sec. 3.2 for details on how this is implemented. To train μ_θ, we define the *forward process* as a Markov chain:

$$q(x_{1:T}^{0:N} \mid x_0^{0:N}) = \prod_{t=1}^{T} \prod_{n=0}^{N} q(x_t^n \mid x_{t-1}^n), \quad (2)$$

where $q(x_t^n \mid x_{t-1}^n) = \mathcal{N}(x_t^n; \sqrt{1-\beta_t} x_{t-1}^n, \beta_t \mathbf{I})$ and $\beta_1, \ldots, \beta_T$ define a constant variance schedule, i.e., we apply separate noise per image to produce training samples.

We follow Ho et al. [14] by learning a *noise predictor* ϵ_θ instead of μ_θ. This allows to train ϵ_θ with an $L2$ loss:

$$\mathbb{E}_{x_0^{0:N}, \epsilon^{0:N} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), n} \left[\left\| \epsilon^n - \epsilon_\theta^n(x_t^{0:N}, t) \right\|^2 \right]. \quad (3)$$

3.2. Augmentation of the U-Net architecture

To model a 3D-consistent denoising process over all images, we predict per-image noise $\epsilon_\theta^n(x_t^{0:N}, t)$ through a neural network ϵ_θ. This neural network is initialized from the pretrained weights of existing text-to-image models, and is usually defined as a U-Net architecture [31, 35]. We seek to leverage the previous states $x_t^{0:N}$ of all images to arrive at a 3D-consistent denoising step. To this end, we propose to add two layers into the U-Net architecture, namely a cross-frame-attention layer and a projection layer. We note that the predicted per-image noise needs to be image specific, since

all images are generated starting from separate Gaussian noise. It is therefore important to keep around 2D layers that act separately on each image, which we achieve by finetuning the existing ResNet [11] and ViT [9] blocks. We summarize our architecture in Fig. 2. In the following, we discuss our two proposed layers in more detail.

Cross-Frame Attention. Inspired by video diffusion [54, 56], we add cross-frame-attention layers into the U-Net architecture. Concretely, we modify the existing self-attention layers to calculate $\mathit{CFAttn}(Q,K,V)=\mathit{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$ with

$$Q = W^Q h_i,\ K = W^K [h_j]_{j\neq i},\ V = W^V [h_j]_{j\neq i}, \quad (4)$$

where W^Q, W^K, W^V are the pretrained weights for feature projection, and $h_i \in \mathbb{R}^{C\times H\times W}$ is the input spatial feature of each image $i \in [1, N]$. Intuitively, this matches features across all frames, which allows generating the same global style.

Additionally, we add a conditioning vector to all cross-frame and cross-attention layers to inform the network about the viewpoint of each image. First, we add pose information by encoding each image's camera matrix $p \in \mathbb{R}^{4\times4}$ into an embedding $z_1 \in \mathbb{R}^4$ similar to Zero-1-to-3 [24]. Additionally, we concatenate the focal length and principal point of each camera into an embedding $z_2 \in \mathbb{R}^4$. Finally, we provide an intensity encoding $z_3 \in \mathbb{R}^2$, which stores the mean and variance of the image RGB values. At training time, we set z_3 to the true values of each input image, and at test time, we set $z_3=[0.5, 0]$ for all images. This helps to reduce the view-dependent lighting differences contained in the dataset (e.g., due to different camera exposure). We construct the conditioning vector as $z=[z_1, z_2, z_3]$, and add it through a LoRA-linear-layer [16] W'^Q to the feature projection matrix Q. Concretely, we compute the projected features as:

$$Q = W^Q h_i + s \cdot W'^Q [h_i; z], \quad (5)$$

where we set $s=1$. Similarly, we add the condition via W'^K to K, and W'^V to V.

Projection Layer. Cross-frame attention layers are helpful to produce globally 3D-consistent images. However, the objects do not precisely follow the specified poses, which leads to view-inconsistencies (see Fig. 5 and Tab. 3). To this end, we add a projection layer into the U-Net architecture (Fig. 3). The idea of this layer is to create 3D-consistent features that are then further processed by the next U-Net layers (e.g. ResNet blocks). By repeating this layer across all stages of the U-Net, we ensure that the per-image features are in a 3D-consistent space. We do not add the projection layer to the first and last U-Net blocks, as we saw no benefit from them at these locations. We reason that the network processes image-specific information at those stages and thus does not need a 3D-consistent feature space.

Inspired by multi-view stereo literature [3, 17, 41], we create a 3D feature voxel grid from all input spatial features

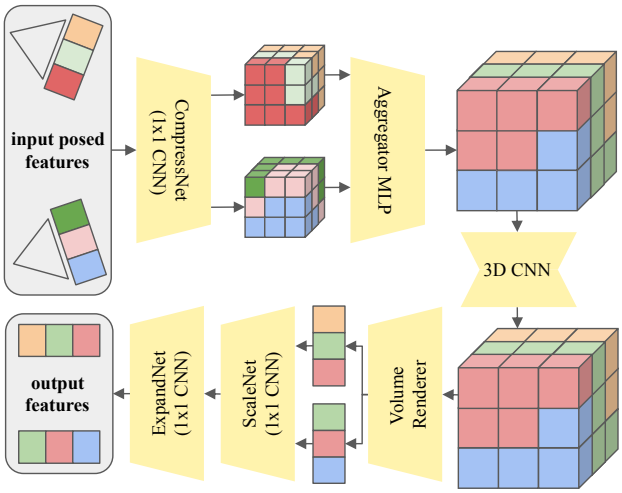


Figure 3. **Architecture of the projection layer.** We produce 3D-consistent output features from posed input features. First, we unproject the compressed image features into 3D and aggregate them into a joint voxel grid with an MLP. Then we refine the voxel grid with a 3D CNN. A volume renderer similar to NeRF [26] renders 3D-consistent features from the grid. Finally, we apply a learned scale function and expand the feature dimension.

$h_{\text{in}}^{0:N} \in \mathbb{R}^{C\times H\times W}$ by projecting each voxel into each image plane. First, we compress $h_{\text{in}}^{0:N}$ with a 1×1 convolution to a reduced feature dimension $C'=16$. We then take the bilinearly interpolated feature at the image plane location and place it into the voxel. This way, we create a separate voxel grid per view, and merge them into a single grid through an aggregator MLP. Inspired by IBRNet [51], the MLP predicts per-view weights followed by a weighted feature average. We then run a small 3D CNN on the voxel grid to refine the 3D feature space. Afterwards, we render the voxel grid into output features $h_{\text{out}}^{0:N} \in \mathbb{R}^{C'\times H\times W}$ with volumetric rendering similar to NeRF [26]. We dedicate half of the voxel grid to foreground and half to background and apply the background model from MERF [32] during ray-marching.

We found it is necessary to add a scale function after the volume rendering output. The volume renderer typically uses a *sigmoid* activation function as the final layer during ray-marching [26]. However, the input features are defined in an arbitrary floating-point range. To convert $h_{\text{out}}^{0:N}$ back into the same range, we non-linearly scale the features with 1×1 convolutions and *ReLU* activations. Finally, we expand $h_{\text{out}}^{0:N}$ to the input feature dimension C. We refer to the supplemental material for details about each component's architecture.

3.3. Autoregressive Generation

Our method takes as input multiple samples $x_t^{0:N}$ at once and denoises them 3D-consistently. During training, we set $N=5$, but can increase it at inference time up to memory constraints, e.g., $N=30$. However, we want to render an

object from *any* possible viewpoint directly with our network. To this end, we propose an autoregressive image generation scheme, i.e., we condition the generation of next viewpoints on previously generated images. We provide the timesteps $t^{0:N}$ of each image as input to the U-Net. By varying $t^{0:N}$, we can achieve different types of conditioning.

Unconditional Generation. All samples are initialized to Gaussian noise and are denoised jointly. The timesteps $t^{0:N}$ are kept identical for all samples throughout the *reverse process*. We provide different cameras per image and a single text prompt. The generated images are 3D-consistent, showing the object from the desired viewpoints (Figs. 4 and 5).

Image-Conditional Generation. We divide the total number of samples $N{=}n_\text{c}{+}n_\text{g}$ into a conditional part n_c and generative part n_g. The first n_c samples correspond to images and cameras that are provided as input. The other n_g samples should generate novel views that are similar to the conditioning images. We start the generation from Gaussian noise for the n_g samples and provide the un-noised images for the other samples. Similarly, we set $t^{0:n_\text{c}}{=}0$ for all denoising steps, while gradually decreasing $t^{n_\text{g}:N}$.

When $n_\text{c}{=}1$, our method performs single-image reconstruction (Fig. 6). Setting $n_\text{c}{>}1$ allows to autoregressively generate novel views from previous images (Fig. 1 bottom). In practice, we first generate one batch of images unconditionally and then condition the next batches on a subset of previous images. This allows us to render smooth trajectories around 3D objects (see the supplemental material).

3.4. Implementation Details

Dataset. We train our method on the large-scale CO3Dv2 [33] dataset, which consists of posed multi-view images of real-world objects. Concretely, we choose the categories `Teddybear`, `Hydrant`, `Apple`, and `Donut`. Per category, we train on 500–1000 objects with each 200 images at resolution 256×256. We generate text captions with the BLIP-2 model [21] and sample one of 5 proposals per object.

Training. We base our model on a pretrained latent-diffusion text-to-image model. We only fine-tune the U-Net and keep the VAE encoder and decoder frozen. In each iteration, we select $N{=}5$ images and their poses. We sample one denoising timestep $t{\sim}[0, 1000]$, add noise to the images according to Eq. 2, and compute the loss according to Eq. 3. In the projection layers, we skip the last image when building the voxel grid, which enforces to learn a 3D representation that can be rendered from novel views. We train our method by varying between unconditional and image-conditional generation (Sec. 3.3). Concretely, with probabilities $p_1{=}0.25$ and $p_2{=}0.25$ we provide the first and/or second image as input and set the respective timestep to zero. Similar to Ruiz et al. [34], we create a *prior dataset* with the pretrained text-to-image model and use it during training to maintain the 2D prior (see supplemental material for details).

We fine-tune the model on 2× A100 GPUs for 60K iterations (7 days) with a total batch size of 64. We set the learning rate for the volume renderer to 0.005 and for all other layers to $5{\times}10^{-5}$, and use the AdamW optimizer [35]. During inference, we can increase N and generate up to 30 images/batch on an RTX 3090 GPU. We use the UniPC [61] sampler with 10 denoising steps, which takes 15 seconds.

Table 1. **Quantitative comparison of unconditional image generation.** We report average FID [12] and KID [2] per category and improve by a significant margin. This signals that our images are more similar to the distribution of real images in the dataset. We mask away the background for our method and the real images to ensure comparability of numbers with the baselines.

Category	HF [18]		VD [42]		Ours	
	FID↓	KID↓	FID↓	KID↓	FID↓	KID↓
Teddybear	81.93	0.072	201.71	0.169	**49.39**	**0.036**
Hydrant	61.19	0.042	138.45	0.118	**46.45**	**0.033**
Donut	105.97	0.091	199.14	0.136	**68.86**	**0.054**
Apple	62.19	0.056	183.67	0.149	**56.85**	**0.043**

4. Results

Baselines. We compare against recent state-of-the-art works for 3D generative modeling. Our goal is to create multi-view consistent images from real-world, realistic objects with authentic surroundings. Therefore, we consider methods that are trained on real-world datasets and select HoloFusion (HF) [18], ViewsetDiffusion (VD) [42], and DFM [46]. We show results on two tasks: unconditional generation (Sec. 4.1) and single-image reconstruction (Sec. 4.2).

Metrics. We report FID [12] and KID [2] as common metrics for 2D/3D generation and measure the multi-view consistency of generated images with peak signal-to-noise ratio (PSNR), structural similarity index (SSIM), and LPIPS [60]. To ensure comparability, we evaluate all metrics on images without backgrounds, as not every baseline models them.

4.1. Unconditional Generation

Our method can be used to generate 3D-consistent views of an object from any pose with only text as input by using our autoregressive generation (Sec. 3.3). Concretely, we sample an (unobserved) image caption from the test set for the first batch and generate $N{=}10$ images with a guidance scale [13] of $\lambda_\text{cfg}{=}7.5$. We then set $\lambda_\text{cfg}{=}0$ for subsequent batches, and create a total of 100 images per object.

We evaluate against HoloFusion (HF) [18] and ViewsetDiffusion (VD) [42]. We report quantitative results in Tab. 1 and qualitative results in Figs. 4 and 5. HF [18] creates diverse images that sometimes show view-dependent floating

Figure 4. **Unconditional image generation of our method and baselines.** We show renderings from different viewpoints for multiple objects and categories. Our method produces consistent objects and backgrounds. Our textures are sharper in comparison to baselines. Please see the supplemental material for more examples and animations.

artifacts (see Fig. 5). VD [42] creates consistent but blurry images. In contrast, our method produces images with backgrounds and higher-resolution object details. Please see the suppl. material for more examples and animated results.

4.2. Single-Image Reconstruction

Our method can be conditioned on multiple images in order to render any novel view in an autoregressive fashion (Sec. 3.3). To measure the 3D-consistency of our generated images, we compare single-image reconstruction against ViewsetDiffusion (VD) [42] and DFM [46]. Concretely, we sample one image from the dataset and generate 20 images at novel views also sampled from the dataset. We follow Szymanowicz et al. [42] and report the per-view maximum PSNR/SSIM and average LPIPS across multiple objects and viewpoints for all methods. We report quantitative results in

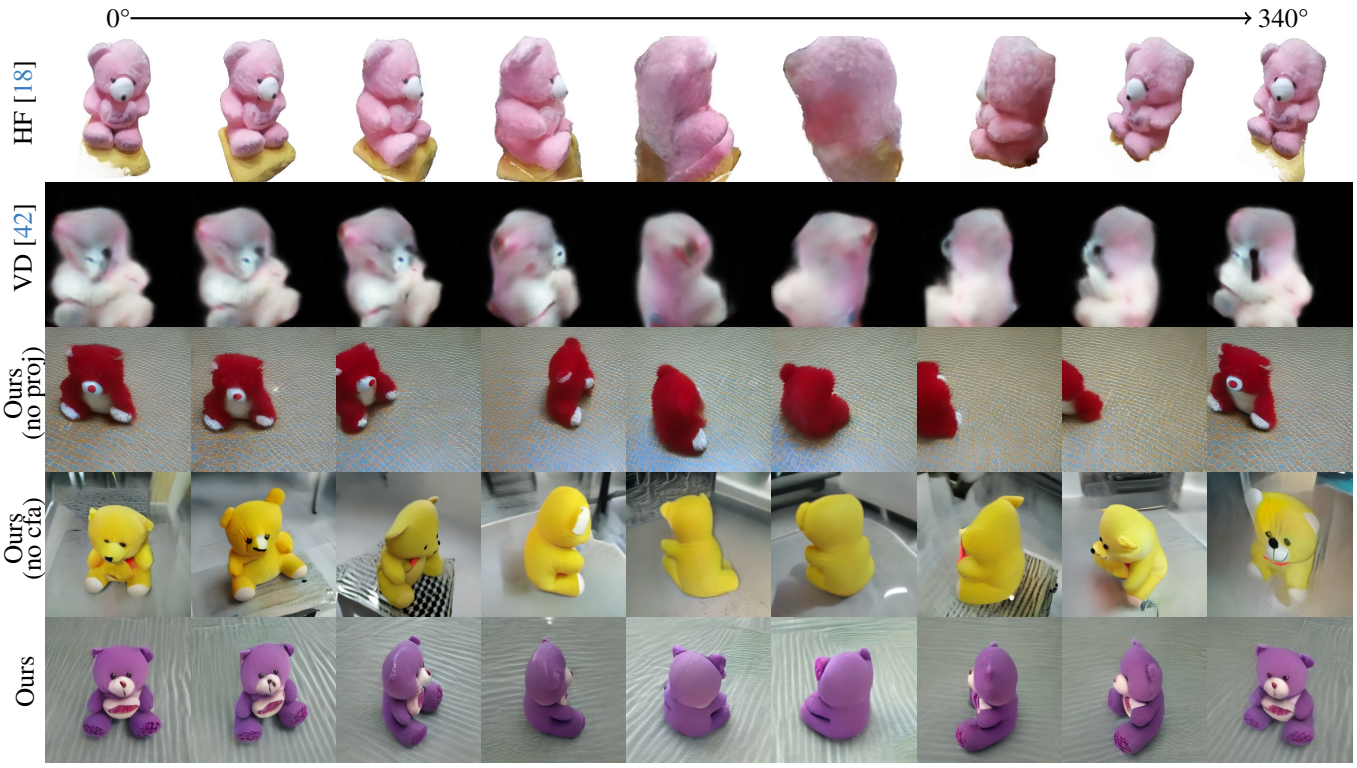


Figure 5. **Multi-view consistency of unconditional image generation.** HoloFusion (HF) [18] has view-dependent floating artifacts (the base in first row). ViewsetDiffusion (VD) [42] has blurrier renderings (second row). Without the projection layer, our method has no precise control over viewpoints (third row). Without cross-frame-attention, our method suffers from identity changes of the object (fourth row). Our full method produces detailed images that are 3D-consistent (fifth row).

Table 2. **Quantitative comparison of single-image reconstruction.** Given a single image as input, we measure the quality of novel views through average PSNR, SSIM, and LPIPS [60] per category. We mask away the generated backgrounds to ensure comparability across all methods. We improve over VD [42] while being on-par with DFM [46].

Method	Teddybear			Hydrant			Donut			Apple		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
VD [42]	19.68	0.70	0.30	22.36	0.80	0.19	18.27	0.68	0.14	19.54	0.64	0.31
DFM [46]	21.81	0.82	0.16	**22.67**	0.83	0.12	**23.91**	**0.86**	**0.10**	25.79	**0.91**	**0.07**
Ours	**21.98**	**0.84**	**0.13**	22.49	**0.85**	**0.11**	21.50	0.85	0.18	**25.94**	**0.91**	0.11

Tab. 2 and show qualitative results in Fig. 6. VD [42] creates plausible results without backgrounds. DFM [46] creates consistent results with backgrounds at a lower image resolution (128×128). Our method produces higher resolution images with similar reconstruction results and backgrounds as DFM [46]. Please see the supplemental material for more examples and animated results.

4.3. Ablations

The key ingredients of our method are the cross-frame-attention and projection layers that we add to the U-Net (Sec. 3.2). We highlight their importance in Tab. 3 and Fig. 5.

How important are the projection layers? They are necessary to allow precise control over the image viewpoints (e.g., Fig. 5 row 3 does not follow the specified rotation). Our goal is to generate a consistent set of images from *any* viewpoint *directly* with our model (Sec. 3.3). Being able to control the pose of the object is therefore an essential part of our contribution. The projection layers build up a 3D representation of the object that is explicitly rendered into 3D-consistent features through volume rendering. This allows us to achieve viewpoint consistency, as also demonstrated through single-image reconstruction (Tab. 3).

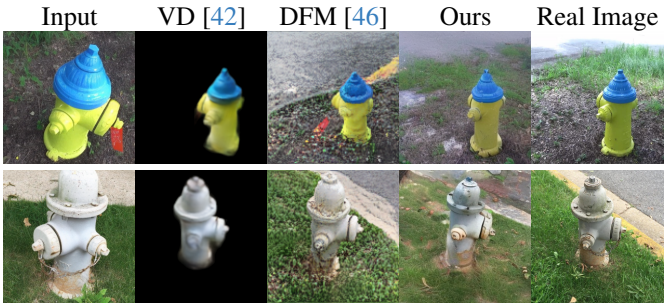


Figure 6. **Single-image reconstruction of our method and baselines.** Given one image/pose as input, our method produces plausible novel views that are consistent with the real shape and texture. We can also produce detailed backgrounds that match the input.

Figure 7. **Diversity of generated results.** We condition our method on text input, which allows to create objects in a desired style. We show samples for hand-crafted text descriptions that combine attributes (e.g., color, shape, background) in a novel way. Each row shows a different generation proposal from our method and we denote the object category (`Teddybear`, `Hydrant`) as `[C]`. This showcases the diversity of generated results, i.e., multiple different objects are generated for the same description.

How important are cross-frame-attention layers? They are necessary to create images of the same object. Without them, the teddybear in Fig. 5 (row 4) has the same general color scheme and follows the specified poses. However, differences in shape and texture lead to an inconsistent set of images. We reason that the cross-frame-attention layers are essential for defining a consistent *object identity*.

Table 3. **Quantitative comparison of our method and ablations.** We report average PSNR, SSIM, LPIPS [60], FID [12], and KID [2] over the `Teddybear` and `Hydrant` categories. We compare against dropping the projection layer ('Ours no proj') and the cross-frame-attention layer ('Ours no cfa') from the U-Net (see Sec. 3.2). While still producing high-quality images with similar FID/KID scores, this demonstrates that our proposed layers are necessary to obtain 3D-consistent images.

Method	PSNR↑	SSIM↑	LPIPS↓	FID↓	KID↓
Ours (no proj)	16.55	0.71	0.29	47.95	**0.034**
Ours (no cfa)	18.15	0.76	0.25	47.93	**0.034**
Ours	**22.24**	**0.84**	**0.11**	**47.92**	**0.034**

Does the 2D prior help? We utilize a 2D prior in form of the pretrained text-to-image model that we fine-tune in a 3D-consistent fashion (Sec. 3.1). This enables our method to produce sharp and detailed images of objects from different viewpoints. Also, we train our method on captioned images to retain the controllable generation through text descriptions (Sec. 3.4). We show the diversity and controllability of our generations in Fig. 7 with hand-crafted text prompts. This highlights that, after finetuning, our model is still faithful to text input, and can combine attributes in a novel way, i.e., our model learns to extrapolate from the training set.

4.4. Limitations

Our method generates 3D-consistent, high-quality images of diverse objects according to text descriptions or input images. Nevertheless, there are several limitations. First, our method sometimes produces images with slight inconsistency, as shown in the supplement. Since the model is fine-tuned on a real-world dataset consisting of view-dependent effects (e.g., exposure changes), our framework learns to generate such variations across different viewpoints. A potential solution is to add lighting condition through a ControlNet [59]. Second, our work focuses on objects, but similarly scene-scale generation on large datasets [6, 57] can be explored.

5. Conclusion

We presented ViewDiff, a method that, given text or image input, generates 3D-consistent images of real-world objects that are placed in authentic surroundings. Our method leverages the expressivity of large 2D text-to-image models and fine-tunes this 2D prior on real-world 3D datasets to produce diverse multi-view images in a joint denoising process. The core insight of our work are two novel layers, namely cross-frame-attention and the projection layer (Sec. 3.2). Our autoregressive generation scheme (Sec. 3.3) allows to directly render high-quality and realistic novel views of a generated 3D object.

6. Acknowledgements

This work was done during Lukas' internship at Meta Reality Labs Zurich as well as at TU Munich, funded by a Meta sponsored research agreement. Matthias Nießner was also supported by the ERC Starting Grant Scan2CAD (804724). We also thank Angela Dai for the video voice-over.

References

[1] Titas Anciukevičius, Zexiang Xu, Matthew Fisher, Paul Henderson, Hakan Bilen, Niloy J Mitra, and Paul Guerrero. Renderdiffusion: Image diffusion for 3D reconstruction, inpainting and generation. In *CVPR*, 2023. 2

[2] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *ICLR*, 2018. 5, 8, 12

[3] Aljaž Božič, Pablo Palafox, Justus Thies, Angela Dai, and Matthias Nießner. TransformerFusion: Monocular RGB scene reconstruction using transformers. In *NeurIPS*, 2021. 4

[4] Eric R. Chan, Koki Nagano, Matthew A. Chan, Alexander W. Bergman, Jeong Joon Park, Axel Levy, Miika Aittala, Shalini De Mello, Tero Karras, and Gordon Wetzstein. Generative novel view synthesis with 3D-aware diffusion models. arXiv:2304.02602, 2023. 2

[5] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3D: Disentangling geometry and appearance for high-quality text-to-3D content creation. In *ICCV*, 2023. 1, 2

[6] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, 2017. 8

[7] 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 *CVPR*, 2023. 2, 15

[8] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *NeurIPS*, 2021. 2

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16×16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 4

[10] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-NeRF2NeRF: Editing 3D scenes with instructions. In *ICCV*, 2023. 2

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4, 13, 15

[12] 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. In *NeurIPS*, 2017. 5, 8, 12

[13] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS Workshops*, 2021. 2, 5, 12, 15

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 2, 3

[15] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2Room: Extracting textured 3D meshes from 2D text-to-image models. In *ICCV*, 2023. 2

[16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022. 4

[17] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *JMLR*, 24(251):1–43, 2023. 4

[18] Animesh Karnewar, Niloy J. Mitra, Andrea Vedaldi, and David Novotny. HoloFusion: Towards photo-realistic 3D generative modeling. In *ICCV*, 2023. 2, 5, 6, 7, 12, 15

[19] Animesh Karnewar, Andrea Vedaldi, David Novotny, and Niloy J. Mitra. HoloDiffusion: Training a 3D diffusion model using 2D images. In *CVPR*, 2023. 2, 12

[20] Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. Learning blind video temporal consistency. In *European Conference on Computer Vision*, 2018. 15

[21] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. arXiv:2301.12597, 2023. 5, 12

[22] 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 *CVPR*, 2023. 2

[23] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Zexiang Xu, Hao Su, et al. One-2-3-45: Any single image to 3D mesh in 45 seconds without per-shape optimization. arXiv:2306.16928, 2023. 2

[24] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9298–9309, 2023. 2, 4, 15

[25] 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. In *The Twelfth International Conference on Learning Representations*, 2024. 2, 3, 15

[26] 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. 2, 4, 15, 20, 21, 22

[27] 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), 2022. 20, 21, 22

[28] Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, and Matthias Nießner. DiffRF:

Rendering-guided 3D radiance field diffusion. In *CVPR*, 2023. 2

[29] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. In *ICLR*, 2023. 1, 2, 15

[30] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, et al. Magic123: One image to high-quality 3D object generation using both 2D and 3D diffusion priors. arXiv:2306.17843, 2023. 2

[31] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. arXiv:2204.06125, 2022. 1, 2, 3

[32] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. MERF: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 42(4):1–12, 2023. 4, 15

[33] 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 *ICCV*, 2021. 2, 3, 5, 12

[34] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. DreamBooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*, 2023. 1, 5, 12

[35] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022. 1, 2, 3, 5

[36] Nikita Selin. CarveKit. github.com/OPHoperHPO/image-background-remove-tool, 2023. 13, 20, 22

[37] Junyoung Seo, Wooseok Jang, Min-Seop Kwak, Jaehoon Ko, Hyeonsu Kim, Junho Kim, Jin-Hwa Kim, Jiyoung Lee, and Seungryong Kim. Let 2D diffusion model know 3D-consistency for robust text-to-3D generation. arXiv:2303.07937, 2023. 2

[38] Yichun Shi, Peng Wang, Jianglong Ye, Long Mai, Kejie Li, and Xiao Yang. MVDream: Multi-view diffusion for 3D generation. arXiv:2308.16512, 2023. 2

[39] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015. 3

[40] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 15

[41] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. NeuralRecon: Real-time coherent 3D reconstruction from monocular video. In *CVPR*, 2021. 4

[42] Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. ViewSet diffusion: (0-)image-conditioned 3D generative models from 2D data. In *ICCV*, 2023. 2, 5, 6, 7, 8, 12, 13, 15

[43] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *SIGGRAPH*, 2023. 20

[44] Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen. Make-it-3D: High-fidelity 3D creation from a single image with diffusion prior. In *ICCV*, 2023. 2

[45] Shitao Tang, Fuyang Zhang, Jiacheng Chen, Peng Wang, and Yasutaka Furukawa. MVDiffusion: Enabling holistic multi-view image generation with correspondence-aware diffusion. In *NeurIPS*, 2023. 2

[46] Ayush Tewari, Tianwei Yin, George Cazenavette, Semon Rezchikov, Joshua B. Tenenbaum, Frédo Durand, William T. Freeman, and Vincent Sitzmann. Diffusion with forward models: Solving stochastic inverse problems without direct supervision. In *NeurIPS*, 2023. 2, 5, 6, 7, 8, 13, 15

[47] Christina Tsalicoglou, Fabian Manhardt, Alessio Tonioni, Michael Niemeyer, and Federico Tombari. TextMesh: Generation of realistic 3D meshes from text prompts. In *3DV*, 2024. 2

[48] Hung-Yu Tseng, Qinbo Li, Changil Kim, Suhib Alsisan, Jia-Bin Huang, and Johannes Kopf. Consistent view synthesis with pose-guided diffusion models. In *CVPR*, 2023. 2

[49] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score Jacobian chaining: Lifting pretrained 2D diffusion models for 3D generation. In *CVPR*, 2023. 2

[50] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021. 2, 20, 22

[51] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. IBRNet: Learning multi-view image-based rendering. In *CVPR*, 2021. 4, 14

[52] 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. arXiv:2305.16213, 2023. 1, 2, 15

[53] Daniel Watson, William Chan, Ricardo Martin Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. Novel view synthesis with diffusion models. In *ICLR*, 2023. 2

[54] Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Weixian Lei, Yuchao Gu, Wynne Hsu, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. In *ICCV*, 2023. 4

[55] Jianfeng Xiang, Jiaolong Yang, Binbin Huang, and Xin Tong. 3D-aware image generation using 2D diffusion models. In *ICCV*, 2023. 2

[56] Shuai Yang, Yifan Zhou, Ziwei Liu, , and Chen Change Loy. Rerender a video: Zero-shot text-guided video-to-video translation. In *SIGGRAPH Asia*, 2023. 4

[57] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A high-fidelity dataset of 3D indoor scenes. In *ICCV*, 2023. 8

[58] Zehao Yu, Anpei Chen, Bozidar Antic, Songyou Peng, Apratim Bhattacharyya, Michael Niemeyer, Siyu Tang, Torsten Sattler, and Andreas Geiger. SDFStudio: A unified framework for surface reconstruction, 2023. 20

[59] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023. 1, 2, 8, 13

[60] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5, 7, 8, 13

[61] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. UniPC: A unified predictor-corrector framework for fast sampling of diffusion models. In *NeurIPS*, 2023. 5

[62] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Transactions on Graphics*, 37(4):65:1–12, 2018. 2

[63] Zhizhuo Zhou and Shubham Tulsiani. SparseFusion: Distilling view-conditioned diffusion for 3D reconstruction. In *CVPR*, 2023. 2

[64] Joseph Zhu and Peiye Zhuang. HiFA: High-fidelity text-to-3D with advanced diffusion guidance. arXiv:2305.18766, 2023. 2

ViewDiff: 3D-Consistent Image Generation with Text-to-Image Models

Supplementary Material

A. Supplemental Video

Please watch our attached video [1] for a comprehensive evaluation of the proposed method. We include rendered videos of multiple generated objects from novel trajectories at different camera elevations (showcasing unconditional generation as in Fig. 4). We also show animated results for single-image reconstruction (Fig. 6) and sample diversity (Fig. 7).

B. Training Details

B.1. Data Preprocessing

We train our method on the large-scale CO3Dv2 [33] dataset, which consists of posed multi-view images of real-world objects. Concretely, we choose the categories `Teddybear`, `Hydrant`, `Apple`, and `Donut`. Per category, we train on 500–1000 objects, with 200 images at resolution 256×256 for each object. We generate text captions with the BLIP-2 model [21] and sample one of 5 proposals per object during each training iteration. With probability p_1=0.5 we select the training images randomly per object and with probability p_2=0.5 we select consecutive images from the captured trajectory that goes around the object. We randomly crop the images to a resolution of 256×256 and normalize the camera poses such that the captured object lies in an axis-aligned unit cube. Specifically, we follow Szymanowicz et al. [42] and calculate a rotation transformation such that all cameras align on an axis-aligned plane. Then, we translate and scale the camera positions, such that their bounding box is contained in the unit cube.

B.2. Prior Preservation Loss

Inspired by Ruiz et al. [34], we create a *prior preservation* dataset of 300 images and random poses per category with the pretrained text-to-image model. We use it during training to maintain the image generation prior. This has been shown to be successful when fine-tuning a large 2D diffusion model on smaller-scale data [34]. For each of the 300 images we randomly sample a text description from the training set of CO3Dv2 [33]. We then generate an image with the pretrained text-to-image model given that text description as input. During each training iteration we first calculate the diffusion objective (Eq. 3) on the N=5 multi-view images sampled from the dataset and obtain L_d. Then, we sample one image of the *prior preservation* dataset and apply noise to it (Eq. 2). Additionally, we sample a camera (pose and intrinsics) that lies within the distribution of cameras for each object category. We then similarly calculate the loss (Eq. 3) on the prediction of our model and obtain L_p. Since we only sample a single image instead of multiple, this does not train the diffusion model on 3D-consistency. Instead, it trains the model to maintain its image generation prior. Concretely, the cross-frame-attention layers are treated again as self-attention layers and the projection layers perform unprojection and rendering normally, but only from a single image as input. In practice, we scale the prior preservation loss with factor 0.1 and add it to the dataset loss to obtain the final loss: L=$L_d + 0.1L_p$.

C. Evaluation Details

C.1. Autoregressive Generation

We showcase unconditional generation of our method in Sec. 4.1. To obtain these results, we employ our autoregressive generation scheme (Sec. 3.3). Concretely, we sample an (unobserved) image caption from the test set for the first batch and generate N=10 images with a guidance scale [13] of λ_{cfg}=7.5. Then we set λ_{cfg}=0 for subsequent batches and create a total of 100 images per object. We found that the results are most consistent, if the first batch generates N images in a 360° rotation around the object. This way, we globally define the object shape and texture in a single denoising forward pass. All subsequent batches are conditioned on all N images of the first batch. To render a smooth trajectory, we sample the camera poses in other batches in a sequence. That is, the next N images are close to each other with only a small rotation between them. We visualize this principle in our supplemental video.

C.2. Metric Computation

We give additional details on how we computed the metrics as shown in Tabs. 1 to 3. To ensure comparability, we evaluate all metrics on images without backgrounds as not every baseline models them.

FID/KID. We report FID [12] and KID [2] as common metrics for 2D/3D generation. We calculate these metrics to compare *unconditional* image generation against HoloFusion [18] and ViewsetDiffusion [42]. This quantifies the similarity of the generated images to the dataset and thereby provides insight about their quality (e.g., texture details and sharpness) and diversity (e.g., different shapes and colors). Following the baselines [18, 19], we sample 20,000 images from the CO3Dv2 [33] dataset for each object category. We remove the background from each object by using the foreground mask probabilities contained in the dataset. Similarly, we generate 20,000 images with each method and remove the

[1] https://youtu.be/SdjoCqHzMMk

Figure 8. **Limitations.** Our method generates consistent images at different camera poses. However, there can be slight inconsistencies like different sharpness and lighting between images. Since our model is fine-tuned on a real-world dataset consisting of view-dependent effects (e.g., exposure changes), our framework learns to generate such variations across different viewpoints.

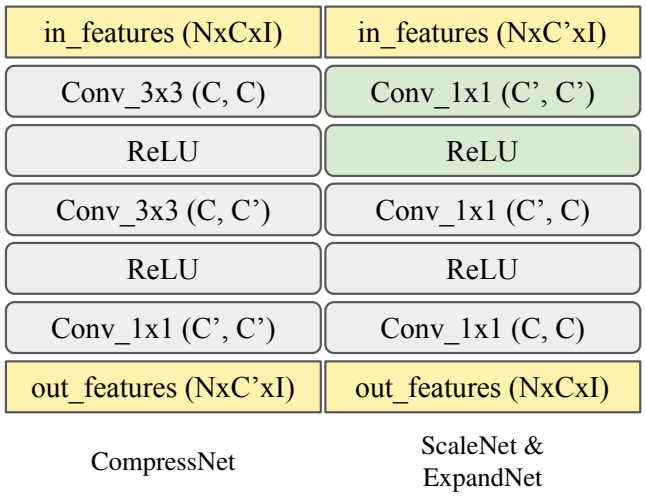


Figure 9. **Architecture of projection layer components.** The projection layer contains the components *CompressNet*, *ScaleNet* (green), and *ExpandNet*. We implement these networks as small CNN networks.

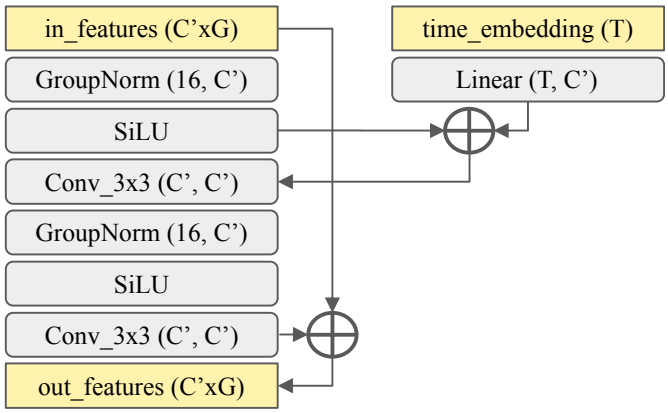


Figure 10. **Architecture of projection layer components.** The projection layer contains the component *3D CNN*. We implement this networks as a series of 5 3D ResNet [11] blocks with timestep embeddings.

background from our generated images with CarveKit [36]. For our method, we set the text prompt to an empty string during the generation to facilitate complete unconditional generation.

PSNR/SSIM/LPIPS. We measure the multi-view consistency of generated images with peak signal-to-noise ratio (PSNR), structural similarity index (SSIM), and LPIPS [60]. We calculate these metrics to compare single-image reconstruction against ViewsetDiffusion [42] and DFM [46]. We resize all images to the resolution 256×256 to obtain comparable numbers. First, we obtain all objects that were not used during training for every method (hereby obtaining a unified test set across all methods). Then, we randomly sample 20 posed image pairs from each object. We use the first image/pose as input and predict the novel view at the second pose. We then calculate the metrics as the similarity of the prediction to the ground-truth second image. We remove the background from the prediction and ground-truth images by obtaining the foreground mask with CarveKit [36] from the *prediction* image. We use the same mask to remove background from both images. This way, we calculate the metrics only on similarly masked images. If the method puts the predicted object at a wrong position, we would thus quantify this as a penalty by comparing the segmented object to the background of the ground-truth image at that location.

D. Limitations

Our method generates 3D-consistent, high-quality images of diverse objects according to text descriptions or input images. Nevertheless, there are several limitations. Our method sometimes produces images with slight inconsistency, as shown in Fig. 8. Since the model is fine-tuned on a real-world dataset consisting of view-dependent effects (e.g., exposure changes), our framework learns to generate such variations across different viewpoints. This can lead to flickering artifacts when rendering a smooth video from a generated set of images (e.g., see the supplemental video). A potential solution is to (i) filter blurry frames from the dataset, and (ii) add lighting-condition through a ControlNet [59].

E. Projection Layer Architecture

We add a projection layer into the U-Net architecture of pre-trained text-to-image models (see Fig. 3 and Sec. 3.2). The idea of this layer is to create 3D-consistent features that are

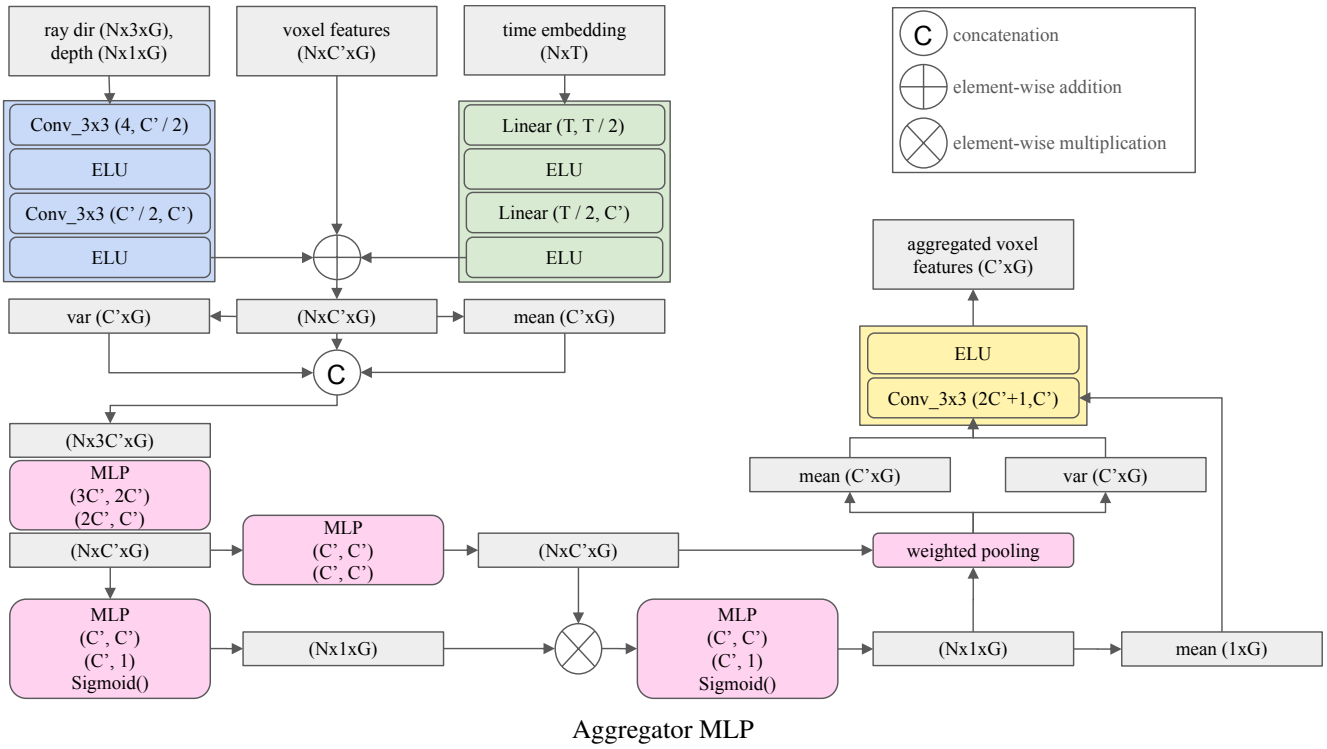


Figure 11. **Architecture of projection layer components.** The projection layer contains the component *Aggregator MLP*. First, we combine per-view voxel grids with their ray-direction/depth encodings (blue) and the temporal embedding (green). Inspired by IBRNet [51], the MLPs (pink) then predict per-view weights followed by a weighted feature average. Finally, we combine the per-voxel weights with the mean and variance grids (yellow) to obtain the aggregated feature grid.

then further processed by the next U-Net layers (e.g. ResNet blocks). Concretely, we create a 3D representation from all input features in form of a voxel grid, that is defined inside of the axis-aligned unit cube. We set the 3D feature dimension as $C' = 16$ and define the base resolution of the voxel grid as $128 \times 128 \times 128$. Throughout the U-Net, we apply the same up/downsampling as for the 2D features, i.e., the resolution decreases to $8 \times 8 \times 8$ in the bottleneck layer. The projection layer consists of multiple network components. We show detailed network architectures of these components in Figs. 9 to 11.

E.1. CompressNet and ExpandNet

We apply the 3D layers on features that are defined in a unified dimensionality of C'=16. Since our 3D layers act on dense voxel grids this helps to lower the memory requirements. To convert to/from this compressed feature space, we employ small CNNs, as depicted in Fig. 9. In these schematics, we define N as the number of images in a batch, C as the uncompressed feature dimension and I as the spatial dimension of the features.

E.2. Aggregator MLP

After creating per-view voxel grids via raymarching (see Sec. 3.2), we combine N voxel grids into one voxel grid that represents the features for all viewpoints. To this end, we employ a series of networks, as depicted in Fig. 11. In these schematics, we define N as the number of images in a batch, C' as the compressed feature dimension, T as the dimension of the timestep embedding, G as the 3D voxel grid resolution, and I as the spatial dimension of the features. The MLPs are defined as a sequence of linear layers of specified input and output dimensionality with *ELU* activations in between.

First, we concatenate the voxel features with an encoding of the ray-direction and depth that was used to project the image features into each voxel. We also concatenate the timestep embedding to each voxel. This allows to combine per-view voxel grids of different timesteps (e.g., as proposed in image conditional generation in Sec. 3.3). It is also useful to inform the subsequent networks about the denoising timestep, which allows to perform the aggregation differently throughout the denoising process. Inspired by IBRNet [51], a set of MLPs then predict per-view weights followed by a weighted feature average. We perform this averaging operation elementwise: since all voxel grids are defined in

the same unit cube, we can combine the same voxel across all views. Finally, we combine the per-voxel weights with the mean and variance grids to obtain the final aggregated feature grid.

E.3. 3D CNN

After aggregating the per-view voxel grids into a joint grid, we further refine that grid. The goal of this network is to add additional details to the feature representation such as the global orientation of the shape. To achieve this, we employ a series of 5 3D ResNet [11] blocks with timestep embeddings, as depicted in Fig. 10. In these schematics, we define C' as the compressed feature dimension, T as the dimension of the timestep embedding, and G as the 3D voxel grid resolution.

E.4. Volume Renderer and ScaleNet

After we obtain a refined 3D feature representation in form of the voxel grid, we render that grid back into per-view image features (see Fig. 3). Concretely, we employ a volume renderer similar to NeRF [26] and implement it as a grid-based renderer similar to DVGO [40]. This allows to render features in an efficient way that is not a bottleneck for the forward pass of the network. In contrast to NeRF, we render down *features* instead of *rgb* colors. Concretely, we sample 128 points along a ray and for each point we trilinearly interpolate the voxel grid features to obtain a feature vector $f \in \mathbb{R}^{C'}$. Then, we employ a small 3-layer MLP that transforms f into the density $d \in \mathbb{R}$ and a sampled feature $s \in \mathbb{R}^{C'}$. Using alpha-compositing, we accumulate all pairs $(d_0, s_0), ..., (d_{127}, s_{127})$ along a ray into a final rendered feature $r \in \mathbb{R}^{C'}$. We dedicate half of the voxel grid to foreground and half to background and apply the background model from MERF [32] during ray-marching.

We found it is necessary to add a scale function after the volume rendering output. The volume renderer typically uses a *sigmoid* activation function as the final layer during ray-marching [26]. However, the input features are defined in an arbitrary floating-point range. To convert r back into the same range, we non-linearly scale the features with 1×1 convolutions and *ReLU* activations. We depict the architecture of this *ScaleNet* as the green layers in Fig. 9.

F. Additional Results

F.1. Comparison To Additional Baselines

We compare against additional text-to-3D baselines that also utilize a pretrained text-to-image model in Fig. 12. We choose ProlificDreamer [52] as representative of score distillation [29] methods. Rendered images are less photorealistic since the optimization may create noisy surroundings and over-saturated textures. Similar to us, Zero123-XL [24] and SyncDreamer [25] circumvent this problem by generating 3D-consistent images directly. However, they finetune on a large synthetic dataset [7] instead of real-world images. As a result, their images have synthetic textures and lighting effects and no backgrounds. We quantify this in Tab. 4 with the FID between sets of generated images (conditioned on an input view), and real images of the same object (without backgrounds). Our method has better scores since the generated images are more photorealistic.

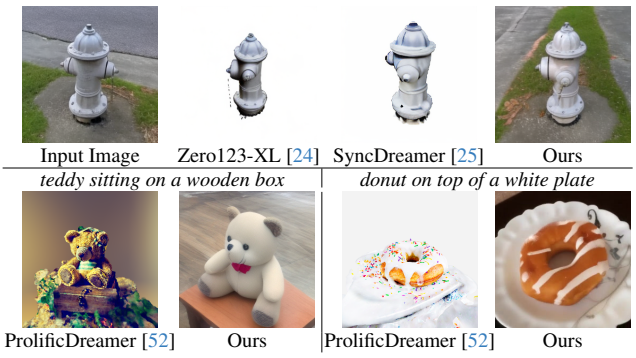


Figure 12. **Comparison to other text-to-3D baselines from image- (top) and text-input (bottom).** Our method produces images with higher photorealism and authentic surroundings.

Table 4. **Comparison of consistency (mid) and photorealism (FID).** Our method shows similar 3D-consistency as baselines, while producing more photorealistic images.

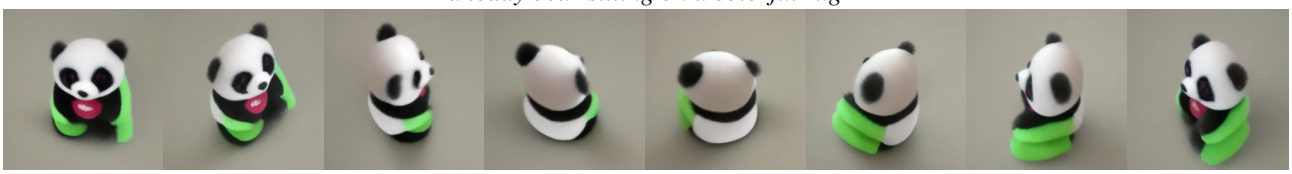
Method	$E_{\text{warp}} \downarrow$	#Points↑	PSNR↑	FID↓
DFM [46]	0.0034	17,470	32.32	—
VD [42]	0.0021	—	—	—
HF [18]	0.0031	—	—	—
SyncDreamer [25]	0.0042	4,126	33.81	135.78
Zero123-XL (SDS) [24]	0.0039	—	—	126.83
Ours	0.0036	18,358	33.65	85.08

We calculate temporal stability (E_{warp}) of video renderings with optical flow warping following [20]. Also, we measure the consistency of generated images for methods that do not directly produce a 3D representation. Concretely, we report the number of point correspondences following [25] and the PSNR between NeRF [26] re-renderings and input images. Table 4 shows that our method is on-par with baselines in terms of 3D consistency, while generating higher quality images.

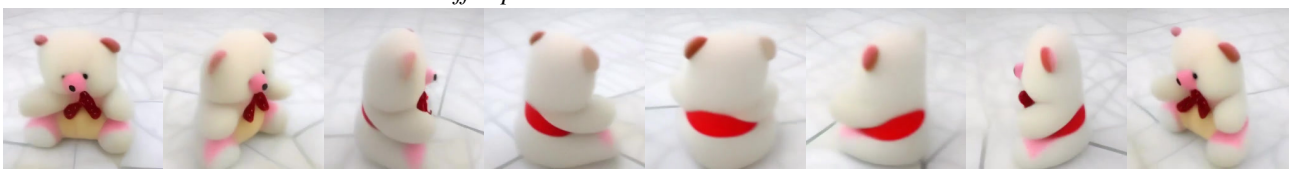
F.2. Unconditional Generation

We generate images in a similar fashion as in Sec. 4.1. Concretely, we sample an (unobserved) image caption from the test set for the first batch and generate N=10 images with a guidance scale [13] of λ_{cfg}=7.5. Then we set λ_{cfg}=0 for subsequent batches and create a total of 100 images per object. We show additional results in Figs. 13 to 16.

a teddy bear sitting on a colorful rug

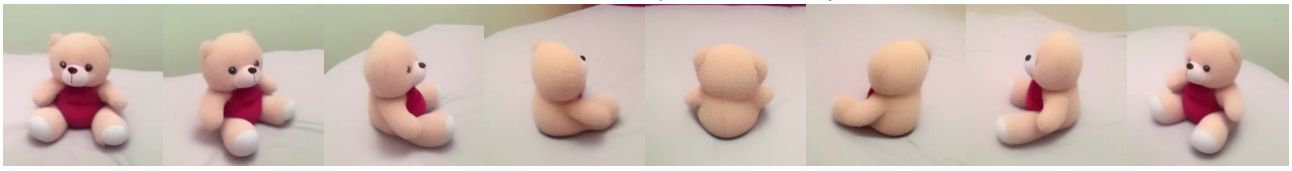


a stuffed panda bear with a heart on its chest

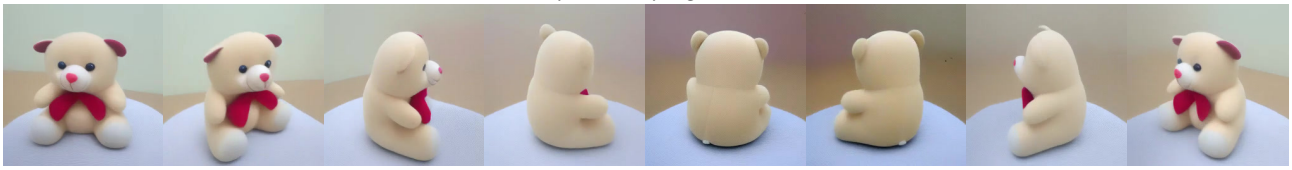


a stuffed animal sitting on a tile floor

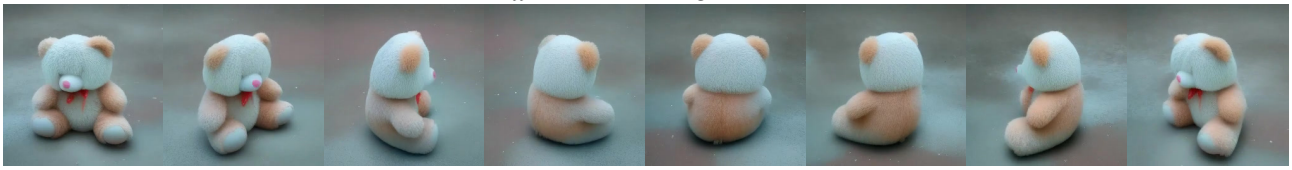
a black and white teddybear with blue feet



a teddy bear laying on a bed



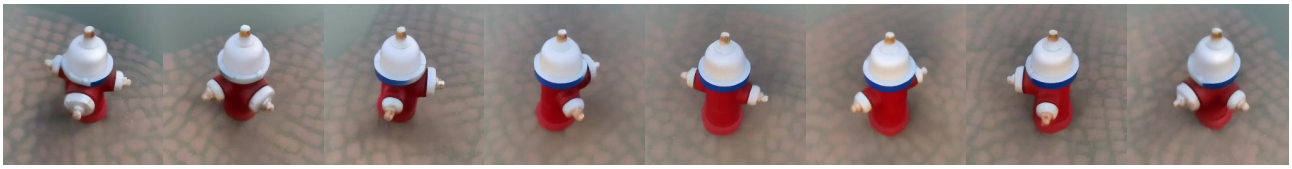
a stuffed animal sitting on a chair



a teddy bear sitting on the ground in the dark

a stuffed bear wearing a red hat and a cloak

Figure 13. **Additional examples of our method.** Given a text prompt as input, we generate a smooth trajectory around an object with our autoregressive generation scheme (Sec. 3.3). Please see the supplemental video for animations of the generated samples.

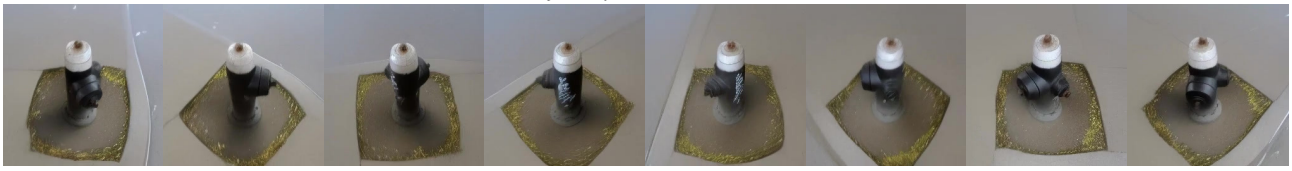


a red and white fire hydrant on a brick floor

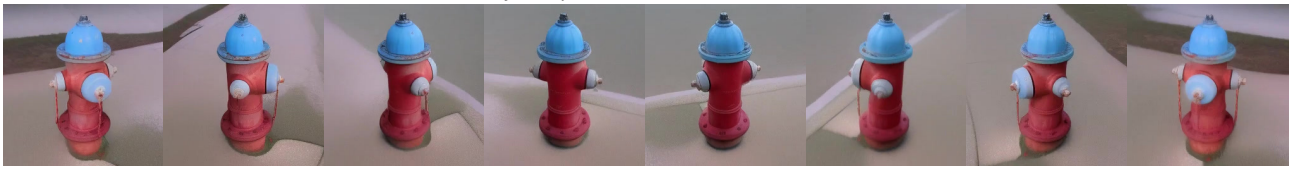
a yellow and green fire hydrant sitting on the ground

a yellow fire hydrant sitting on the sidewalk

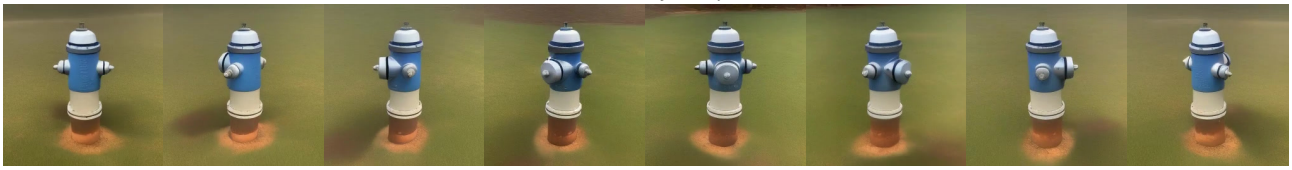
a red fire hydrant in the snow



a fire hydrant on the sidewalk



a red and blue fire hydrant



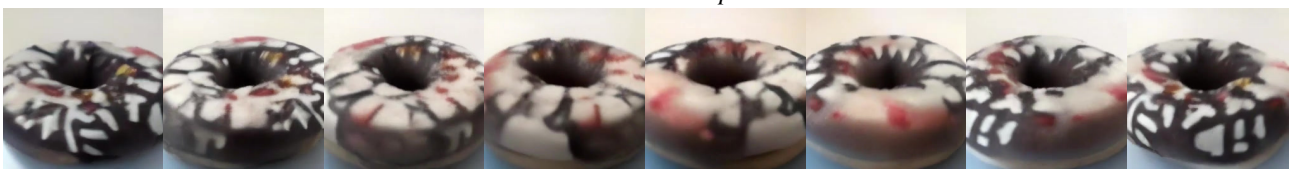
a blue and white fire hydrant sitting in the grass

a green fire hydrant with a tag on it

Figure 14. **Additional examples of our method.** Given a text prompt as input, we generate a smooth trajectory around an object with our autoregressive generation scheme (Sec. 3.3). Please see the supplemental video for animations of the generated samples.

a glazed donut sitting on a marble counter

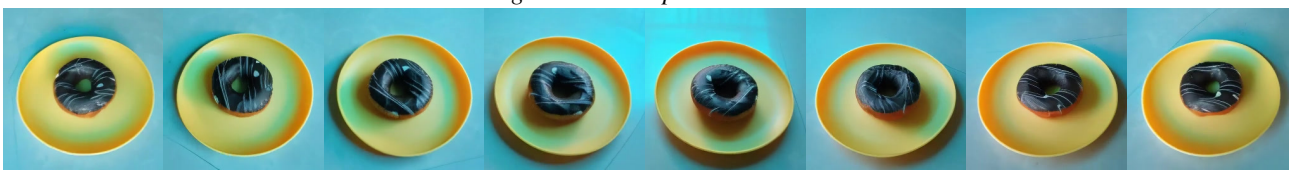
a donut on a clear plate



a chocolate donut with sprinkles on it

a donut on a plate with a hole in it

a large donut on a plate on a table



a yellow plate with a donut on it

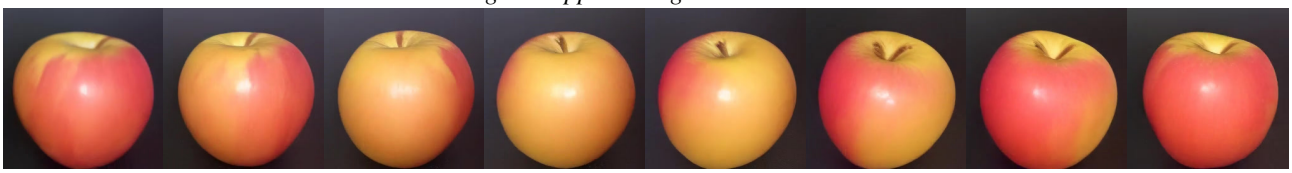
a white plate with a donut on it

a donut sitting on a cloth

Figure 15. **Additional examples of our method.** Given a text prompt as input, we generate a smooth trajectory around an object with our autoregressive generation scheme (Sec. 3.3). Please see the supplemental video for animations of the generated samples.

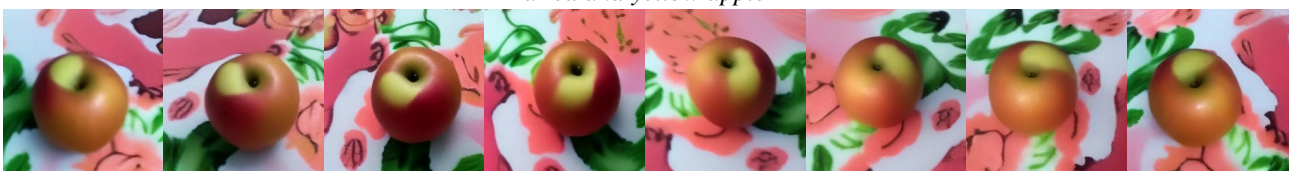
a red apple on a white counter top

a green apple sitting on a counter

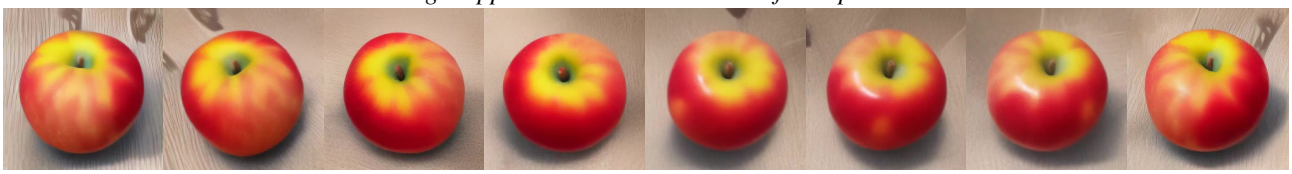


a red and yellow apple

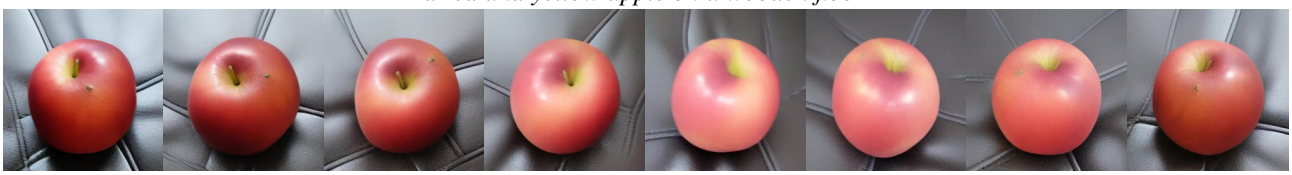
a red and yellow apple



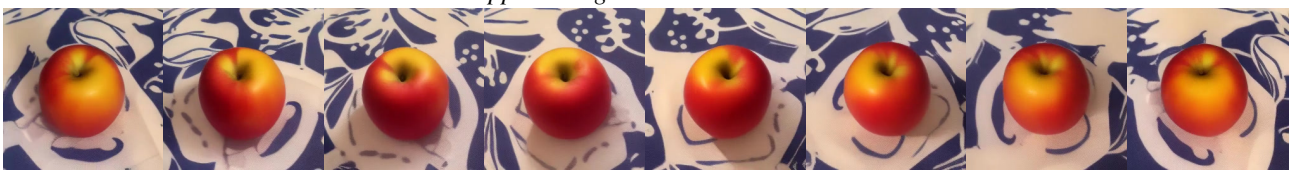
a single apple on a table cloth with a floral pattern



a red and yellow apple on a wooden floor



a red apple sitting on a black leather couch



a red apple on a blue and white patterned table cloth

Figure 16. **Additional examples of our method.** Given a text prompt as input, we generate a smooth trajectory around an object with our autoregressive generation scheme (Sec. 3.3). Please see the supplemental video for animations of the generated samples.

G. Optimizing a NeRF/NeuS

Our method is capable of directly rendering images from novel camera positions in an autoregressive generation scheme (see Sec. 3.3). This allows to render smooth trajectories around the same 3D object at arbitrary camera positions. Depending on the use case, it might be desirable to obtain an explicit 3D representation of a generated 3D object (instead of using our method to autoregressively render new images). We demonstrate that our generated images can be used directly to optimize a NeRF [26] or NeuS [50]. Concretely, we optimize a NeRF with the Instant-NGP [27] implementation from our generated images for 10K iterations (2 minutes). Also, we extract a mesh by optimizing a NeuS with the *neus-facto* implementation from SDFStudio [43, 58] for 20K iterations (15 minutes). First, we remove the background of our generated images by applying Carvekit [36] and then start the optimization with these images. We show results in Figs. 17 to 19.

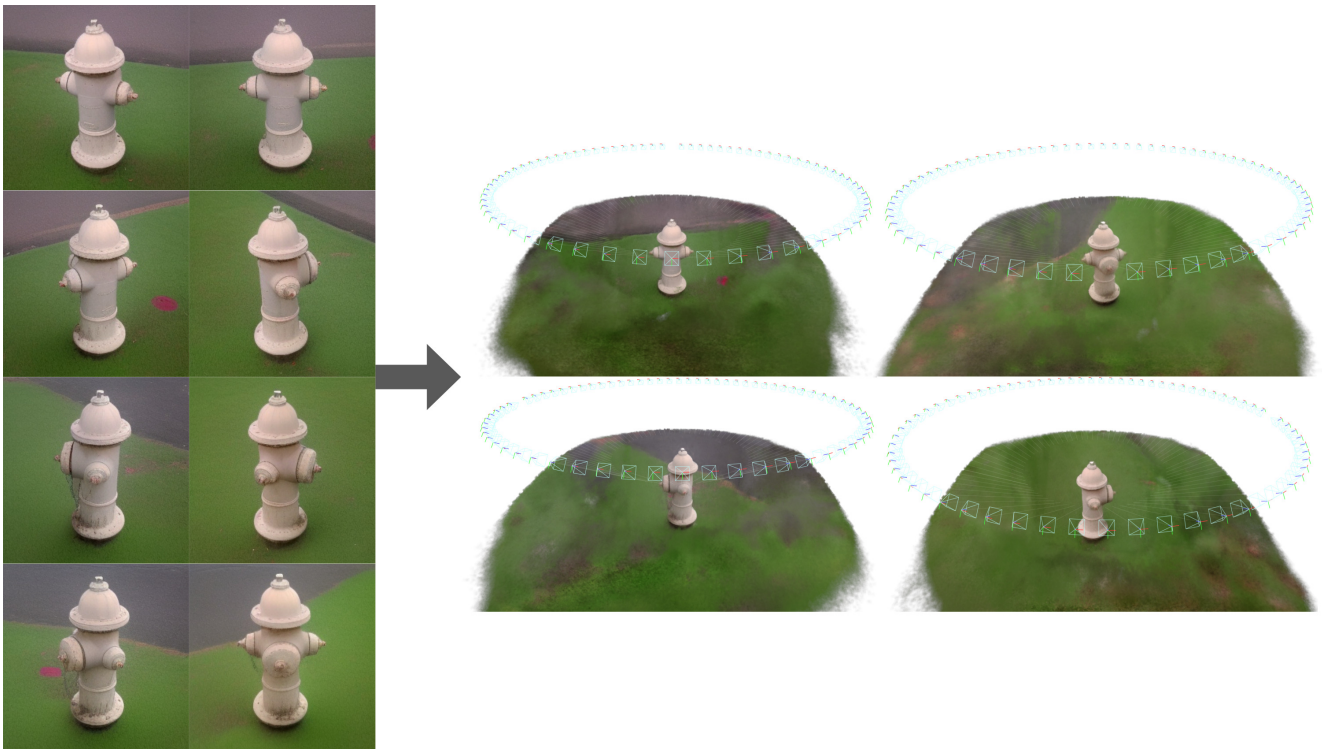


Figure 17. **NeRF [26] optimization from our generated images.** Left: given a text prompt as input, we generate a smooth trajectory around an object with our autoregressive generation scheme (Sec. 3.3). In total, we generate 100 images at different camera positions. Right: we create a NeRF using Instant-NGP [27] from the generated images. We show the camera positions of the generated images on top of the optimized radiance field.

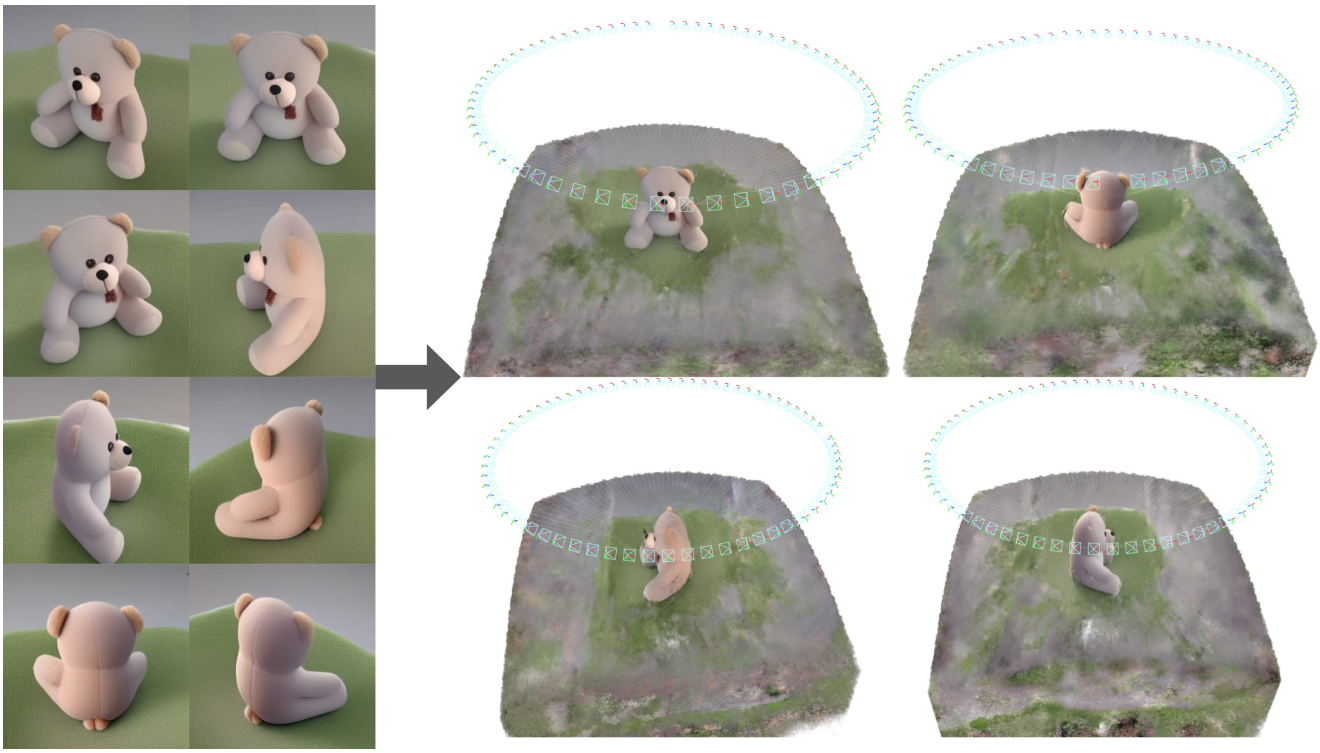


Figure 18. **NeRF [26] optimization from our generated images.** Left: given a text prompt as input, we generate a smooth trajectory around an object with our autoregressive generation scheme (Sec. 3.3). In total, we generate 100 images at different camera positions. Right: we create a NeRF using Instant-NGP [27] from the generated images. We show the camera positions of the generated images on top of the optimized radiance field.

Figure 19. **Mesh extraction from our generated images.** Given a text prompt as input, we generate a smooth trajectory around an object with our autoregressive generation scheme (Sec. 3.3). In total, we generate 100 images at different camera positions and mask-out the background with Carvekit [36]. We then optimize a NeuS [50] and extract the mesh from it (last 4 images per row).