VolumeDiffusion: Feed-forward Text-to-3D Generation with Efficient Volumetric Encoder

Zhicong Tang[1] Shuyang Gu[2] Chunyu Wang[3] Ting Zhang[4]
Jianmin Bao[3] Dong Chen[3] Baining Guo[3]
[1]Tsinghua University [2]University of Science and Technology of China
[3]Microsoft Research Asia [4]Beijing Normal University

Abstract

This work presents VolumeDiffusion, a novel feed-forward text-to-3D generation framework that directly synthesizes 3D objects from textual descriptions. It bypasses the conventional score distillation loss based 3D optimization or text-to-image-to-3D approaches. To scale up the training data for the diffusion model, a novel 3D volumetric encoder is developed to efficiently acquire feature volumes from multi-view images. The 3D volumes are then trained on a diffusion model for text-to-3D generation using a 3D U-Net. This research further addresses the challenges of inaccurate object captions and high-dimensional feature volumes. The proposed model, trained on the public Objaverse dataset, demonstrates promising outcomes in producing diverse and recognizable samples from text prompts. Notably, it empowers finer control over object part characteristics through textual cues, fostering model creativity by seamlessly combining multiple concepts within a single object. This research significantly contributes to the progress of 3D generation by introducing an efficient, flexible, and scalable representation methodology.

Keywords: Text-to-3D, 3D Generation, Diffusion Models

1. Introduction

In recent years, feed-forward generation has proven to be very successful in the fields of text-to-image [46, 43] and text-to-video [42] generation. They have demonstrated their power for the efficient generation of visual content that aligns with textual descriptions. However, the task of text-to-3D generation is mainly dominated by iterative 3D optimization guided by 2D diffusion models [44, 55, 6, 8] or utilizing a sequential text-to-image-to-3D conversion [29, 18, 30, 57] which is a more circuitous generation process as shown in Figure 1.

We believe that the key to unlocking the full potential of feed-forward text-to-3D generation lies in the scalability of training data. To achieve the goal, we need to develop a 3D representation that is *efficient* to compute from the massive data sources such as images and point clouds, and meanwhile *flexible* to interact with text prompts at fine-grained levels. Despite the increasing efforts in 3D generation, the optimal representation for 3D objects remains largely unexplored. Commonly adopted approaches include Tri-plane [54, 16] and implicit neural representations (INRs) [22]. However, Tri-plane have been only validated on objects with limited variations such as human faces and the global representation in INR makes it hard to interact with text prompts.

In this work, we present VolumeDiffusion, a novel feed-forward text-to-3D generation framework. It applies volumetric representation that characterizes both the texture and geometry of small parts of an object using features in each voxel, similar to the concept of pixels in images. Differing from previous approaches such as [32, 5], which require additional images as input, our method allows us to directly render images of target objects using only their feature volumes. Meanwhile, the feature volumes encode generalizable priors from image features, enabling us to use a shared decoder for all objects. The above advantages make the representation well-suited for generation tasks.

To scale up the training data for the subsequent diffusion model, we propose a lightweight network to efficiently acquire feature volumes from multi-view images, bypassing the expensive per-object optimization process required in previous approaches [54]. In our current implementation, this network can process 30 objects per second on a single GPU, allowing us to acquire $500K$ models within hours. It also allows extracting ground-truth volumes on-the-fly for training diffusion models which eliminates the storage overhead associated with feature volumes. In addition to the efficiency, this localized representation also allows for flexible interaction with text prompts at fine-grained object part level. This enhanced controllability paves the way for creative designs by combining a number of concepts in one object.

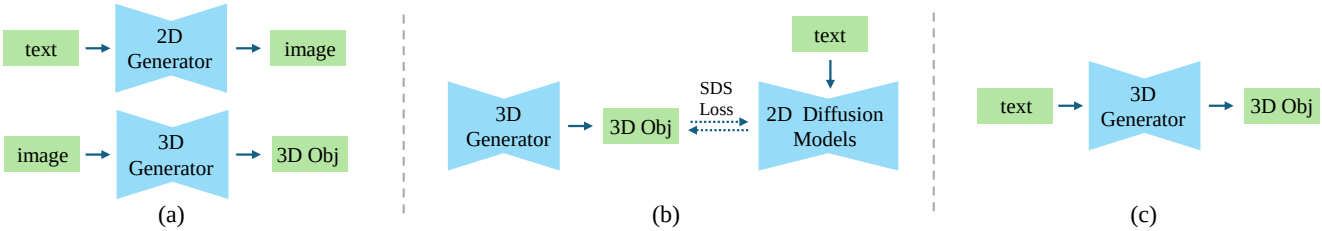


Figure 1. Comparison of different text-to-3D framework. (a) Text-to-image-to-3D approaches which generate 2D image or multi-view images first and then generates 3D data from images, *e.g.*, Instant3d [25], One-2-3-45 [29] and One-2-3-45 [28]. (b) 3D generation guided by 2D diffusion models, *e.g.*, DreamFusion [44] and Fantasia3D [6]. (c) Feed-forward text-to-3D generation (ours) which directly synthesizes 3D objects from textual descriptions.

We train a diffusion model on the acquired 3D volumes for text-to-3D generation using a 3D U-Net [47]. This is a non-trivial task that requires careful design. First, the object captions in the existing datasets [11, 10] are usually inaccurate which may lead to unstable training if not handled properly. To mitigate their adverse effects, we carefully designed a novel schedule to filter out the noisy captions, which notably improves the results. Second, the feature volumes are usually very high-dimensional, *e.g.* $\mathrm{C} \times 32^3$ in our experiments which potentially pose challenges when training the diffusion model. We adopted a new noise schedule that shifted towards larger noise due to increased voxel redundancy. Meanwhile, we proposed the low-frequency noise strategy to effectively corrupt low-frequent information when training the diffusion model. We highlight that this structural noise has even more important effects than that in images due to the higher volume dimension.

We train our model on the public dataset Objaverse [11] which has $800K$ objects ($100K$ after filtering). Our model successfully produces diverse and recognizable samples from text prompts. Compared to Shap·E [22], our model obtains superior results in terms of controlling the characteristics of object parts through text prompts, although we only use less than 10% of the training data (Shap·E trained on several million private data according to their paper). For instance, given the text prompt "*a black chair with red legs*", we observe that Shap·E usually fails to generate red legs. We think it is mainly caused by the global implicit neural representation which cannot interact with text prompts at fine-grained object part level. Instead, our localized volumetric representation, similar to images, can be flexibly controlled by text prompts at voxel level. We believe this is critical to enhance the model's creativity by combining a number of concepts in one object.

2. Related Work

Differentiable Scene Representation. Differentiable scene representation is a class of algorithms that encodes a scene and can be rendered into images while maintaining differentiability. It allows scene reconstruction by optimizing multi-view images and object generation by modeling the representation distribution. It can be divided into implicit neural representation (INR), explicit representation (ER), and hybrid representation (HR).

Neural Radiance Field (NeRF) [35] is a typical INR that encodes a scene as a function mapping from coordinates and view directions to densities and RGB colors. Densities and RGB colors of points along camera rays are integrated to render an image, and the function mapping can be trained to match the ground-truth views. While NeRF uses Multi-Layer Perceptron (MLP) to encode the underlying scene, its success gave rise to many follow-up works that explored different representations.

Plenoxels [14] is an ER that avoids a neural network decoder and directly encodes a scene as densities and spherical harmonic coefficients at each grid voxel. TensoRF [4] further decomposes the voxel grid into a set of vectors and matrices as an ER, and values on each voxel are computed via vector-matrix outer products.

Instant-NGP [38] is an HR that uses a multi-resolution hash table of trainable feature vectors as the input embedding of the neural network decoder and obtains results with fine details. [3] proposed a Tri-plane HR that decomposes space into three orthogonal planar feature maps, and features of points projected to each plane are added together to represent a point in space. DMTet [48] is also an HR that combines a deformable tetrahedral grid and Signed Distance Function (SDF) to obtain a precise shape, and the underlying mesh can be easily exported by Marching Tetrahedra algorithm [12].

Text-to-3D Generation. In Figure 1 we compare three mainstream text-to-3d frameworks. A class of methods first generate 2D image or multi-view images first and then generates 3D data from images [29, 18, 30, 57, 31]. *e.g.*, One-2-3-45++ [28] and LGM [50] generate multi-view images using a fine-tuned 2D diffusion model, and reconstruct a 3D model from them. In comparison, without detailed reference images, our approach is inherently more challenging, but holds the potential to streamline 3D-consistent generations from textual descriptions.

Some recent works escalate to text-to-3D generation via the implicit supervision of pretrained 2D diffusion mod-

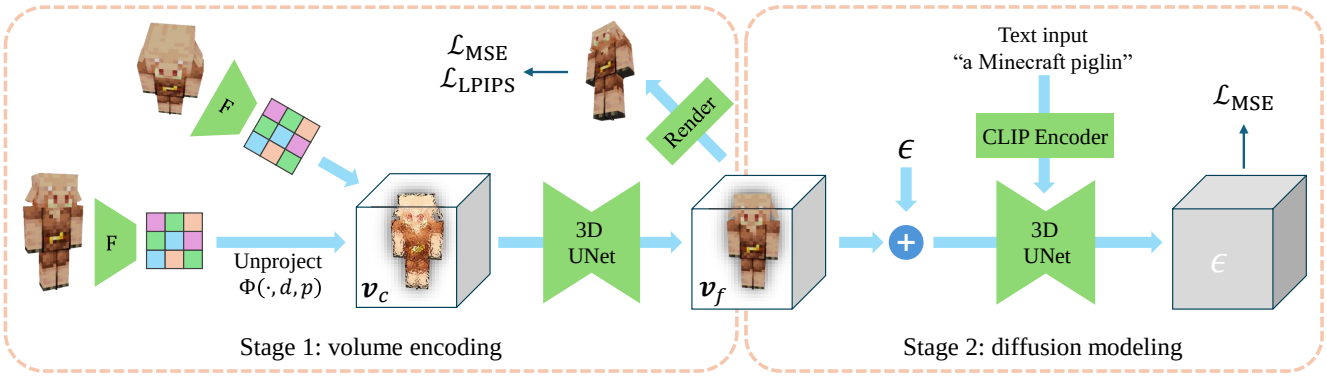


Figure 2. Training framework of VolumeDiffusion. It comprises the volume encoding stage and the diffusion modeling stage. The encoder unprojects multi-view images into a feature volume and do refinements. The diffusion model learns to predict ground-truths given noised volumes and text conditions.

els [44, 55, 6, 8, 34, 51] or vision-language models [21, 36, 56], as shown in Figure 1(b), *e.g.*, DreamField [21] use a contrastive loss of CLIP [45] text feature and rendered image feature to optimize a 3D representation. DreamFusion [44] develop the Score Distillation Sampling (SDS) method, leveraging the semantic understanding and high-quality generation capabilities of text-to-image diffusion models. [52, 58] further combine CLIP, SDS, reference view, and other techniques to push the quality. Though these optimization-based methods yield outstanding visual fidelity and text-3D alignment, they suffer from the cost of time-consuming gradient back-propagation and optimization, which could take hours for each text prompt.

Our work is also related to diffusion-based 3D generation [9, 26, 37, 1, 49, 60], such as DiffTF [2] and Rodin [54] which directly generate 3D representations with diffusion models. However, all of them need optimize and save NeRF parameters in the cost of a time-consuming fitting preparation and expensive storage. In constrast, our volume encoder efficiently acquire feature volumes from multi-view images, bypassing the expensive per-object optimization process.

3. Method

The training of our text-to-3D generation framework comprises two main stages: the encoding of volumes and the diffusion modeling phase. In the volume encoding stage, as discussed in Section 3.1, we have chosen to use feature volume as our 3D representation and utilize a lightweight network to convert multi-view images into 3D volumes. The proposed method is very efficient and bypasses the typically costly optimization process required by previous methods, allowing us to process a substantial number of objects in a relatively short period of time. In the diffusion modeling phase, detailed in Section 3.2, we model the distribution of the previously obtained feature volumes with a text-driven diffusion model. This stage of the process is not without its challenges, particularly in relation to the high dimensionality of the feature volumes and the inaccuracy of object captions in the datasets. We have therefore developed several key designs to mitigate these challenges during the training process.

3.1. Volume Encoder

3.1.1 Volume Representation

One of the key points to train 3D generation models is the selection of appropriate 3D representations to serve as the latent space. The 3D representation should be able to capture the geometry and texture details of the input object and be flexible for fine-grained text control. Furthermore, the 3D representation should be highly efficient in obtaining and reconstructing objects for scalability.

Previous representations such as NeRF [35], Plenoxel [14], DMTet [48], TensoRF [4], Instant-NGP [38], and Tri-plane [3] all have their limitations to serve as the latent space. For instance, the globally shared MLP parameters across different coordinates in NeRFs cause them inflexible and uncontrollable to local changes. Representations storing an explicit 3D grid, like Plenoxel and DMTet, require high spatial resolutions, resulting in large memory costs for detailed scene representation. TensoRF, Instant-NGP, and Tri-plane decompose the 3D grid into multiple sub-spaces with lower dimension or resolution to reduce memory costs but also introduce entanglements.

In this work, we propose a novel representation that merges a lightweight decoder with a feature volume to depict a scene. The lightweight decoder comprises a few layers of MLP, enabling high-resolution, fast, and low-memory cost rendering. The feature volume, instead of storing explicit values, houses implicit features and effectively reduces memory costs. The features of a spatial point are trilinearly interpolated by the nearest voxels on the volume. The decoder inputs the interpolated feature and outputs the density and RGB color of the point. The feature volume is isometric to the 3D space, providing extensive controllability over each part of an object.

3.1.2 Feed-forward Encoder

Unlike previous works [2, 37, 54] that iteratively optimize the representation for each object in a time-consuming way, we use an encoder that directly obtains the feature volume of any object within a forward pass.

As shown in Figure 2, the encoder takes a set of multi-view photos of an object $(\mathbf{x}, \mathbf{d}, \mathbf{p})$, where $\mathbf{x}, \mathbf{d} \in \mathbb{R}^{N\times 3\times H\times W}$ represents the image and depth of N views, $\mathbf{p} = \{p^{(i)}\}_{i=1}^{N}$ represents the corresponding camera parameters, including the camera poses and field of view (FOV). We first extract features from 2D images with a small network $\mathbf{F}$ composed of two layers of convolution. We then unproject the features into a coarse volume $\mathbf{v}_c$ according to depths and camera poses, *i.e.*,

$$\mathbf{v}_c = \Phi(\mathbf{F}(\mathbf{x}), \mathbf{d}, \mathbf{p}), \tag{1}$$

where Φ represents the unproject operation. For each point on camera rays, we first calculate its distance to the camera, then obtain a weight $\mathbf{w}_i = \exp(-\lambda \Delta d_i)$ where Δd_i is the difference of calculated distance and ground-truth depth. The feature of each voxel is the weighted average of features unprojected from different views.

Secondly, we apply a 3D U-Net [47] module to refine the aggregated feature volume to produce a smoother volume

$$\mathbf{v}_f = \Psi(\mathbf{v}_c). \tag{2}$$

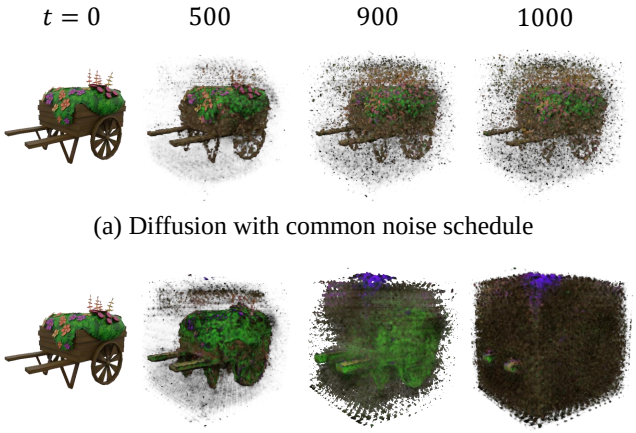
Then ray marching and neural rendering are performed to render images from target views. In the training stage, we optimize the feature extracting network, the 3D U-Net, and the MLP decoder end-to-end with L_2 and LPIPS [59] loss on multi-view rendered images.

The proposed volume encoder is highly efficient for two primary reasons. Firstly, it is capable of generating a high-quality 3D volume with 32 or fewer images once it is trained. This is a significant improvement over previous methods [54], which require more than 200 views for object reconstruction. Secondly, our volume encoder can encode an object in approximately 30 milliseconds using a single GPU. This speed enables us to generate $500K$ models within a matter of hours. As a result, there's no need to store these feature volumes. We extract ground-truth volumes for training diffusion models on-the-fly. It effectively eliminates the expensive storage overhead associated with feature volumes.

3.2. Diffusion Model

3.2.1 Devil in High-dimensional Space

Unlike the conventional text-to-image diffusion models, our text-to-3D diffusion model is designed to learn a latent distribution that is significantly more high-dimensional. This is exemplified in our experiments where we utilize dimensions such as $C \times 32^3$, in stark contrast to the 4×64^2 employed in Stable Diffusion. This heightened dimensionality makes the training of diffusion models more challenging.



(a) Diffusion with common noise schedule

(b) Diffusion with low-frequency noise

Figure 3. Renderings of noised volumes. Volumes with common *i.i.d.* noise are still recognizable at large timesteps, while low-frequency noise effectively removes information.

Figure 3(a) provides illustrations of how noised volumes appear at various timesteps. Utilizing the standard noise schedule employed by Stable Diffusion, the diffusion process cannot effectively corrupt the information. This is evident as the renderings maintain clarity and recognizability, even at large timesteps. Also, it's important to note that there is a huge gap in the information between the noised samples at the final timestep and pure noise. This gap can be perceived as the difference between the training and inference stages. We believe it is due to the high-dimensional character of volume space.

We theoretically analyze the root of this problem. Considering a local patch on the image consisting of $M = w \times h \times c$ values, denoted as $\mathbf{x}_0 = \{x_0^1, x_0^2, \ldots, x_0^M\}$. Without loss of generality, we assume that $\{x_0^i\}_{i=1}^M$ are sampled from Gaussian distribution $\mathcal{N}(0, 1)$. With common strategy, we add *i.i.d.* Gaussian noise $\{\epsilon^i\}_{i=1}^M \sim \mathcal{N}(0, 1)$ to each value by $x_t^i = \sqrt{\bar{\alpha}_t} x_0^i + \sqrt{1-\bar{\alpha}_t}\epsilon^i$ to obtain the noised sample, where $\bar{\alpha}_t$ indicates the noise level at timestep t. Thus the expected mean L_2 perturbation of the patch is

$$\begin{aligned} &\mathbb{E}\left(\frac{1}{M}\sum_{i=0}^{M}\left(x_0^i - x_t^i\right)\right)^2 \\ &= \frac{1}{M^2}\mathbb{E}\left(\sum_{i=0}^{M}\left((1-\sqrt{\bar{\alpha}_t})\mathbf{x}_0^i - \sqrt{1-\bar{\alpha}_t}\epsilon^i\right)\right)^2 \\ &= \frac{2}{M}\left(1-\sqrt{\bar{\alpha}_t}\right). \end{aligned} \tag{3}$$

As the resolution M increases, the *i.i.d.* noises added to each value collectively have a minimal impact on the

patch's appearance, and the disturbance is reduced significantly. The rate of information distortion quickly declines to $\frac{1}{M}$. This observation is consistent with findings from concurrent studies [7, 15, 19]. In order to train diffusion models effectively, it's essential to carefully design an appropriate noise that can distort information. So we propose a new noise schedule and the low-frequency noise in the training process.

3.2.2 New Noise Schedule

The primary goal of our text-to-3D diffusion model is to learn a latent distribution, which is significantly more dimensional than the text-to-image model. As discussed in the previous section, a common noise schedule can lead to insufficient information corruption when applied to high-dimensional spaces, such as volume.

During the training stage, if the information of objects remains a large portion, the network quickly overfits to the noised volumes from the training set and ignores the text conditions. This essentially means that the network leans more towards utilizing information from noised volumes rather than text conditions. To address this, we decided to reduce $\bar{\alpha}_t$ for all timesteps. Thus we reduced the final signal-to-noise ratio from 6×10^{-3} to 4×10^{-4}, and evenly reduced $\bar{\alpha}_t$ at the intermediate timesteps. Without this, the network may fail to output any object when inference from pure Gaussian noise due to the training and inference gap.

We performed a series of experiments using different noise schedules with various hyper-parameters. These includes the commonly used linear [17], cosine [39], and sigmoid [20] schedules. After comprehensive testing and evaluations, we determined the linear noise schedule to be the most suitable for our experiments.

3.2.3 Low-Frequency Noise

Images or feature volumes are typical digital signals, which can be seen as a combination of digital signals of different frequencies. When adding *i.i.d.* Gaussian noise to each voxel of a volume, the signal is essentially perturbed by a white noise. The *i.i.d.* noise evenly corrupts the information of all components through the diffusion process. However, the amplitude of low-frequent components is usually larger and a white noise cannot powerfully corrupt them. Thus, the mean of the whole volume as the component with the lowest frequency is most likely unnoticed during the diffusion process, causing information leaks. And so are patches and structures of different sizes in the volume.

Hence, we proposed the low-frequency noise strategy to effectively corrupt information and train diffusion models. We modulate the high-frequency *i.i.d.* Gaussian noise with an additional low-frequency noise, which is a single value drawn from normal distribution shared by all values in the same channel. Formally, the noise is

$$\epsilon^i = \sqrt{1-\alpha}\,\epsilon_1^i + \sqrt{\alpha}\,\epsilon_2, \tag{4}$$

where $\{\epsilon_1^i\}_{i=1}^M \sim \mathcal{N}(0,1)$ is independently sampled for each location and $\epsilon_2 \sim \mathcal{N}(0,1)$ is shared within the patch. We still add noise to data by $x_t^i = \sqrt{\bar{\alpha}_t}x_0^i + \sqrt{1-\bar{\alpha}_t}\epsilon^i$, but the noise ϵ^i is mixed via Equation 4 and no longer *i.i.d.* .

With the low-frequency noise, the expected mean L_2 perturbation of the patch is

$$\begin{aligned} &\mathbb{E}\left(\frac{1}{M}\sum_{i=0}^{M}\left(x_0^i - x_t^i\right)\right)^2 \\ &= \frac{2}{M}\left(1-\sqrt{\bar{\alpha}_t}\right) + \left(1-\frac{1}{M}\right)(1-\bar{\alpha}_t)\alpha, \end{aligned} \tag{5}$$

where $\alpha \in [0,1]$ is a hyper-parameter. The proof is in the supplemental material. By this approach, we introduce additional information corruption that is adjustable and remains scale as the resolution grows, effectively removing information of objects as shown in Figure 3(b).

The low-frequency noise can be seen as altering the variance of [17] from $\mathbf{I}$ to $\mathbf{\Sigma} = \sqrt{1-\alpha}\mathbf{I} + \sqrt{\alpha}\mathbf{J}$. The closed form of the forward process from x_0 to x_t holds since we can alternatively iterates $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)\mathbf{\Sigma})$ to $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_{t-1}, (1-\bar{\alpha}_t)\mathbf{\Sigma})$, and the estimation of posterior mean in [17] is still valid. We leave the proof in the appendix.

3.3. Refinement

The diffusion model is able to generate a feature volume, but its inherent limitation lies in its output of low-resolution, which restricts texture details. To overcome this, we leveraged existing text-to-image models to generate more detailed textures, enhancing the initial results obtained from the diffusion model.

Specifically, we introduced the third stage involving fine-tuning the results. Given the good initial output from the diffusion model, we incorporated SDS [44] in this stage to optimize results, ensuring better image quality and reduced errors. Considering our initial results are already satisfactory, this stage only requires a few iterations, making our entire process still efficient.

Our methodology makes full use of existing text-to-image models to generate textures that are not covered in the original training set, enhancing the details of texture and promoting diversity in the generated images. Simultaneously, our method also addresses the issue of multiple-face problems encountered in [44].

3.4. Data Filtering

We find that data filtering is extremely important to the training. Objaverse is mainly composed of unfiltered user-uploaded 3D models crawled from the web, including many geometry shapes, planer scans and images, texture-less objects, and flawed reconstruction from images. Moreover, the annotation is usually missing or not related, the rotation and position vary in a wide range, and the quality of 3D models is relatively poor compared to image datasets.

Cap3D [33] propose an approach for automatically generating descriptive text for 3D objects in the Objaverse dataset. They use BLIP-2 [24], a pre-trained vision-language model, to caption multi-view rendered images of one object and summarize them into a final caption with GPT-4 [41]. However, considering the significant variation in captions from different views, even GPT-4 confuses to extract the main concept, hence the final captions are still too noisy for the text-to-3D generation. With these noisy captions, we find that the diffusion model struggles to understand the relation between text conditions and objects.

We generate our own captions with LLaVA [27] and Llama-2 [53] and filter out objects with low-quality or inconsistent multi-view captions in the Objaverse dataset. Similar to Cap3D, we first generate captions of 8 equidistant views around the object and then summarize them into an overall caption with Llama-2. After that, we calculate the similarity matrix of every pair among these 9 captions using CLIP text embedding. We believe that a high-quality 3D object should be visually consistent from different viewpoints, i.e., the captions from different views should be similar. Thus, we use the average and minimal values of the similarity matrix to represent the quality of the object. And manually set two thresholds to filter out objects with low average/minimal similarity scores.

We use a selected subset of objects with the highest quality to train the diffusion model. We find that the diffusion model is able to learn semantics relations from text conditions. On the contrary, when we use the whole Objaverse dataset for training, the model fails to converge.

4. Experiments

4.1. Implementation Details

Dataset We use the Objaverse [11] dataset in our experiments and rendered 40 random views for each object. For the volume encoder, we filter out transparent objects and train with a subset of $750K$ objects. For the diffusion model, we caption and filter as described in Section 3.4 and train with a subset of $100K$ text-object pairs.

Volume encoder In the first stage, we train a volume encoder that efficiently converts multi-view RGBD images into a feature volume. Each image $\mathbf{x}_i$ are fed into a lightweight network $\mathbf{F}$ to extract the feature $\mathbf{F}(\mathbf{x}_i)$. The network $\mathbf{F}$ merely includes 2 layers of 5×5 convolution. Then features of images are unprojected into the coarse volume $\mathbf{v}_c$ and weighted averaged. λ is set to $160N$ in our experiments, where $N = 32$ is the spatial resolution of volume. After unprojection, the volume is refined with a 3D U-Net module and rendered with an MLP. The MLP has 5 layers with a hidden dimension of 64. The volume encoder and the rendering decoder in total have 25M parameters. The model is trained with the Adam [23] optimizer. The learning rate is 10^{-4} for the volume encoder and 10^{-5} for the MLP. The betas are set to $(0.9, 0.99)$ and no weight decay or learning rate decay is applied. The input and rendered image resolution is 256^2 and the batch size of volume is 1 per GPU. We first optimize the model with only L_2 loss on the RGB channel. We randomly select 4096 pixels each from 5 random views as supervision. After $100K$ iterations, we add an additional LPIPS loss with a weight of 0.01. Due to GPU memory limitation, the LPIPS loss is measured on 128×128 patches. The training takes 2 days on 64 V100 GPUs.

Figure 4. Reconstruction results of the volume encoder.

Diffusion model In the second stage, we train a text-conditioned diffusion model to learn the distribution of feature volumes. The denoiser network is a 3D U-Net adopted from [40]. Text conditions are 77×512 embeddings extracted with CLIP ViT-B/32 [13] text encoder and injected into the 3D U-Net with cross-attentions at middle blocks with spatial resolution $\frac{N}{4}$ and $\frac{N}{8}$. We use a linear [17] noise schedule with $T = 1000$ steps and $\beta_T = 0.03$. We train with the proposed low-frequency noise strategy and the noise is mixed via Equation 4 with $\alpha = 0.5$ in our experiments. The model has 340M parameters in total and is

Data	Views	PSNR ↑	SSIM ↑	LPIPS ↓
10K	8	27.14	0.855	0.288
	16	27.50	0.867	0.282
	32	27.61	0.871	0.281
	64	27.64	0.870	0.280
750K (Ours)	32	27.69	0.874	0.279

Table 1. Ablation on input view numbers and training data size of the volume encoder.

optimized with the Adam optimizer. The model is supervised by only L_2 loss on volumes and no rendering loss is applied. The batch size of volume is 24 per GPU, the learning rate is 10^{-5}, the betas are $(0.9, 0.99)$, and the weight decay is 2×10^{-3}. The training takes about 2 weeks on 96 V100 GPUs.

4.2. Volume Encoder

We first quantitatively study the reconstruction quality of the volume encoder. We set the spatial resolution $N = 32$ and channel $C = 4$ for efficiency. In Table 1, we measure the PSNR, SSIM and LPIPS loss between reconstructions and ground-truth images.

To analyze the correlation between the number of different input views and the quality of reconstruction, we train encoders with different input views on a subset of $10K$ data. It is observed that the quality of reconstruction improves as the number of input views increases. However, once the number of input views surpasses 32, the enhancement of quality becomes negligible. Therefore, we opted to use 32 as the default number of input views in our subsequent experiments. Additionally, the quality of reconstruction is also enhanced with the use of more training data.

We show the reconstruction results of the volume encoder in Figure 4. The volume encoder is capable of reconstructing the geometry shape and textures of objects. Experiments involving higher resolution and larger channels will yield more detailed reconstructions. However, these adjustments will also result in increased training costs and complexity in the second stage. Please refer to the supplemental material for additional ablation studies.

4.3. Diffusion Model

We compare our method with other text-to-3D generation approaches, including Shap·E [22], DreamFusion [44], and One-2-3-45 [29]. Since One-2-3-45 is essentially an image-to-3D model, we use images generated with Stable Diffusion as its input.

Figure 5 demonstrates that our methods yield impressive results, whereas both Shap·E and One-2-3-45 struggle to generate complex structures and multiple concepts. For simpler cases, such as a teapot, Shap·E, and One-2-3-45 can only produce a rough geometry, with surfaces not so smooth and continuous as those created by our method. For more complex cases, our model excels at combining multiple objects in a scene and aligning better with the text prompts, whereas other methods can only capture parts of concepts.

Both our method and Shap·E are native methods, *i.e.* directly supervised on 3D representation and trained with 3D datasets. It's noteworthy that these native methods generate clearer and more symmetrical shapes (for example, boxes, planes, and spheres) than methods based on image-to-3D reconstruction or distillation. Furthermore, the results of One-2-3-45 are marred by many white dots and stripes, which we believe is due to the inconsistency between images generated by the pre-trained Zero-1-to-3 [30] model.

In Table 2, we compute the CLIP Similarity and CLIP R-Precision as a quantitative comparison. For each method, we generated 100 objects and rendered 8 views for each object. Our method outperforms others on both visual quality an text alignment.

We present more results in Figure 6. These prompts include cases of concept combinations and attribute bindings. The critical drawbacks of distillation-based methods, including the Janus problem and over-saturated color, are not observed in our results.

Method	Similarity ↑	R-Precision ↑
DreamFusion [44]	0.243	47.3%
One-2-3-45 [29]	0.228	39.1%
Shap-E [22]	0.287	58.9%
Ours	**0.288**	**63.8%**

Table 2. Quantitative comparison with other text-to-3D methods. Similarity and R-Precision are evaluated with CLIP between rendered images and text prompts.

Stage	Method	Time
1 (Encoding)	Fitting	∼35min
	Shap-E [22]	1.2sec
	Ours	**33ms**
2 (Generation)	DreamFusion [44]	∼12hr
	One-2-3-45 [29]	45sec
	Shap-E [22]	14sec
	Ours (w/o refine)	**5sec**
	Ours	∼5min

Table 3. Inference speed comparison. Evaluated on A100 GPU.

4.4. Inference Speed

In Table 3, we report the inference speed of both stages of our method against other approaches. The first stage encodes multi-view images into a 3D representation and is important for scaling up the training data. Shap·E uses a transformer-based encoder that takes both $16K$ point clouds and 20 RGBA images augmented with 3D coordinates as in-

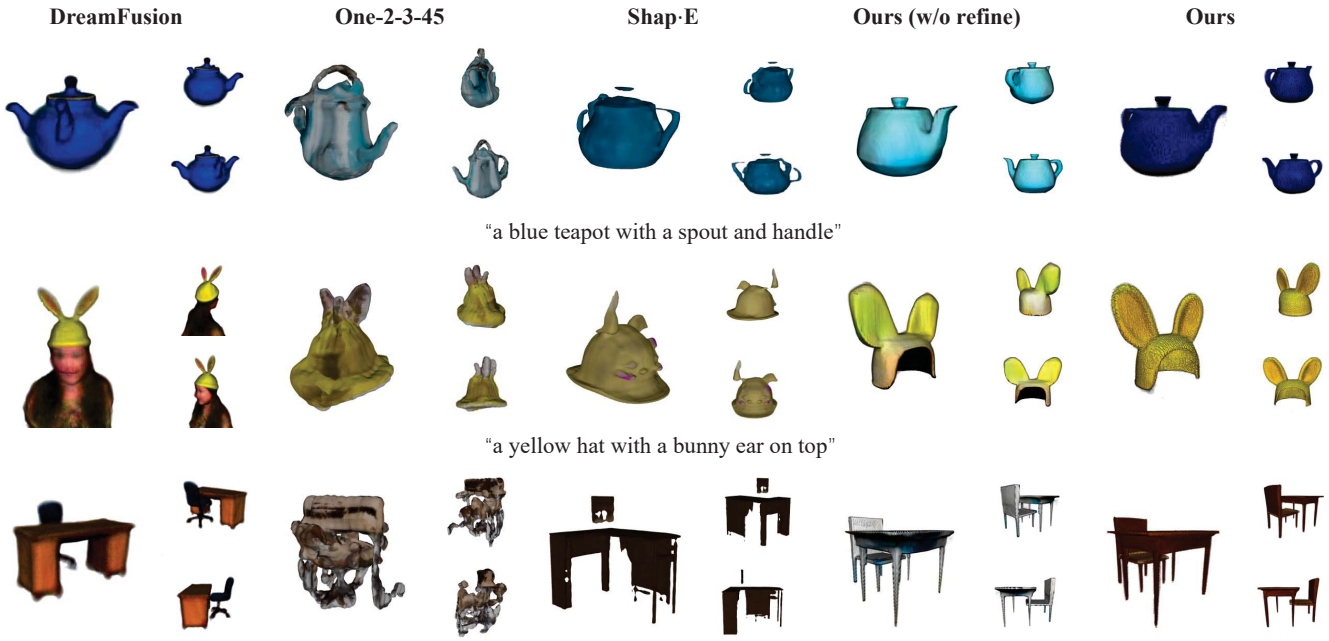


Figure 5. Comparison with other text-to-3D methods.

Method	Similarity ↑	R-Precision ↑
Baseline	0.198	11.3%
+ Low-Frequency Noise	0.201	11.7%
+ New Noise Schedule	0.264	50.7%
+ Both (Ours)	**0.279**	**56.5%**

Table 4. Quantitative comparison between different noise strategy.

put. It is much slower than our lightweight encoder based on convolution. Fitting means to separately optimize a representation for each object with a fixed rendering MLP, and consumes much more time and storage. The second stage refers to the conditional generation process. Optimization-based DreamFusion needs hours for each object. One-2-3-45, on the other hand, necessitates several diffusion-denoising processes, such as text-to-image and multi-view images generation, and is slower than native 3D methods. For both stages, our method proves to be highly efficient.

4.5. Ablation

We conducted ablation experiments on noise schedule and the low-frequency noise in Table 4. We trained diffusion models on a subset of $5K$ data and compares CLIP Similarity and R-Precision. The results demonstrate the effectiveness of our noise strategy.

On noise schedule, we find the baseline method which utilizes the noise schedule employed by Stable Diffusion performs poorly, as the models fail to output any objects while inferencing from pure Gaussian noise. We believe it is due to the information gap between the last timestep and pure noise, which is illustrated in Figure 3(a). Meanwhile, models trained with the proposed new noise schedule eliminate the training-inference gap and are able to draw valid samples from pure noise.

On noise types, we find the model trained with *i.i.d.* noise has lower scores, as it tends to exploit the remaining information of noised volume and confuses when starting from Gaussian noise. In the contrary, the model trained with the low-frequency noise is forced to learn from text conditions and produces results that are more preferable and consistent to text prompts.

5. Conclusion

In conclusion, this paper presented a novel method for efficient and flexible generation of 3D objects from text prompts. The proposed lightweight network for the acquisition of feature volumes from multi-view images has been shown to be an efficient method for scaling up the training data required for the diffusion model. The paper also highlighted the challenges posed by high-dimensional feature volumes and presented a new noise schedule and low-frequency noise for improved the training of diffusion models. In experiments, the superior performance of this model in terms of the control of object characteristics through text prompts has been demonstrated. Our future work would focus on refining the algorithm and the network architecture to further speed up the process. We would also involve testing the model on more diverse datasets, including those with more complex objects and varied text prompts.

Figure 6. Text-to-3D generations by VolumeDiffusion.

References

[1] T. Anciukevičius, Z. Xu, M. Fisher, P. Henderson, H. Bilen, N. J. Mitra, and P. Guerrero. Renderdiffusion: Image diffusion for 3d reconstruction, inpainting and generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12608–12618, 2023. 3

[2] Z. Cao, F. Hong, T. Wu, L. Pan, and Z. Liu. Large-vocabulary 3d diffusion model with transformer. *arXiv preprint arXiv:2309.07920*, 2023. 3, 4

[3] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. De Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16123–16133, 2022. 2, 3

[4] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022. 2, 3

[5] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133, 2021. 1

[6] R. Chen, Y. Chen, N. Jiao, and K. Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. *arXiv preprint arXiv:2303.13873*, 2023. 1, 2, 3

[7] T. Chen. On the importance of noise scheduling for diffusion models. *arXiv preprint arXiv:2301.10972*, 2023. 5, 13

[8] Y. Chen, R. Chen, J. Lei, Y. Zhang, and K. Jia. Tango: Text-driven photorealistic and robust 3d stylization via lighting decomposition. *Advances in Neural Information Processing Systems*, 35:30923–30936, 2022. 1, 3

[9] Y.-C. Cheng, H.-Y. Lee, S. Tulyakov, A. G. Schwing, and L.-Y. Gui. Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4456–4465, 2023. 3

[10] M. Deitke, R. Liu, M. Wallingford, H. Ngo, O. Michel, A. Kusupati, A. Fan, C. Laforte, V. Voleti, S. Y. Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *arXiv preprint arXiv:2307.05663*, 2023. 2

[11] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 2, 6

[12] A. Doi and A. Koide. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE*

TRANSACTIONS on Information and Systems, 74(1):214–224, 1991. 2

[13] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 6

[14] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 2, 3

[15] J. Gu, S. Zhai, Y. Zhang, M. Á. Bautista, and J. M. Susskind. f-dm: A multi-stage diffusion model via progressive signal transformation. In *The Eleventh International Conference on Learning Representations*, 2022. 5

[16] A. Gupta, W. Xiong, Y. Nie, I. Jones, and B. Oğuz. 3dgen: Triplane latent diffusion for textured mesh generation. *arXiv preprint arXiv:2303.05371*, 2023. 1

[17] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 5, 6, 13

[18] Y. Hong, K. Zhang, J. Gu, S. Bi, Y. Zhou, D. Liu, F. Liu, K. Sunkavalli, T. Bui, and H. Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400*, 2023. 1, 2

[19] E. Hoogeboom, J. Heek, and T. Salimans. simple diffusion: End-to-end diffusion for high resolution images. *arXiv preprint arXiv:2301.11093*, 2023. 5

[20] A. Jabri, D. Fleet, and T. Chen. Scalable adaptive computation for iterative generation. *arXiv preprint arXiv:2212.11972*, 2022. 5

[21] A. Jain, B. Mildenhall, J. T. Barron, P. Abbeel, and B. Poole. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 867–876, 2022. 3

[22] H. Jun and A. Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. 1, 2, 7

[23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *The Third International Conference on Learning Representations*, 2015. 6

[24] J. Li, D. Li, S. Savarese, and S. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023. 6

[25] J. Li, H. Tan, K. Zhang, Z. Xu, F. Luan, Y. Xu, Y. Hong, K. Sunkavalli, G. Shakhnarovich, and S. Bi. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. *arXiv preprint arXiv:2311.06214*, 2023. 2

[26] M. Li, Y. Duan, J. Zhou, and J. Lu. Diffusion-sdf: Text-to-shape via voxelized diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12642–12651, 2023. 3

[27] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. In *NeurIPS*, 2023. 6

[28] M. Liu, R. Shi, L. Chen, Z. Zhang, C. Xu, X. Wei, H. Chen, C. Zeng, J. Gu, and H. Su. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10072–10083, 2024. 2

[29] M. Liu, C. Xu, H. Jin, L. Chen, M. Varma T, Z. Xu, and H. Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *Advances in Neural Information Processing Systems*, 36, 2024. 1, 2, 7

[30] R. Liu, R. Wu, B. Van Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9298–9309, 2023. 1, 2, 7

[31] Y. Liu, C. Lin, Z. Zeng, X. Long, L. Liu, T. Komura, and W. Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. *arXiv preprint arXiv:2309.03453*, 2023. 2

[32] X. Long, C. Lin, P. Wang, T. Komura, and W. Wang. Sparseneus: Fast generalizable neural surface reconstruction from sparse views. In *European Conference on Computer Vision*, pages 210–227. Springer, 2022. 1

[33] T. Luo, C. Rockwell, H. Lee, and J. Johnson. Scalable 3d captioning with pretrained models. *arXiv preprint arXiv:2306.07279*, 2023. 6

[34] G. Metzer, E. Richardson, O. Patashnik, R. Giryes, and D. 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. 3

[35] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 2, 3

[36] N. Mohammad Khalid, T. Xie, E. Belilovsky, and T. Popa. Clip-mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia 2022 conference papers*, pages 1–8, 2022. 3

[37] N. Müller, Y. Siddiqui, L. Porzi, S. R. Bulo, P. Kontschieder, and M. Nießner. Diffrf: Rendering-guided 3d radiance field diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4328–4338, 2023. 3, 4

[38] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15, 2022. 2, 3

[39] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 5

[40] A. Q. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. Mcgrew, I. Sutskever, and M. Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *International Conference on Machine Learning*, pages 16784–16804. PMLR, 2022. 6

[41] OpenAI. Gpt-4 technical report, 2023. 6

[42] OpenAI. Video generation models as world simulators, 2024. 1

[43] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023. 1

[44] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2022. 1, 2, 3, 5, 7

[45] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 3

[46] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. 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

[47] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 2, 4

[48] T. Shen, J. Gao, K. Yin, M.-Y. Liu, and S. Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101, 2021. 2, 3

[49] S. Szymanowicz, C. Rupprecht, and A. Vedaldi. Viewset diffusion:(0-) image-conditioned 3d generative models from 2d data. *arXiv preprint arXiv:2306.07881*, 2023. 3

[50] J. Tang, Z. Chen, X. Chen, T. Wang, G. Zeng, and Z. Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. *arXiv preprint arXiv:2402.05054*, 2024. 2

[51] J. Tang, J. Ren, H. Zhou, Z. Liu, and G. Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In *The Twelfth International Conference on Learning Representations*, 2024. 3

[52] J. Tang, T. Wang, B. Zhang, T. Zhang, R. Yi, L. Ma, and D. Chen. Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior. *arXiv preprint arXiv:2303.14184*, 2023. 3

[53] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 6

[54] T. Wang, B. Zhang, T. Zhang, S. Gu, J. Bao, T. Baltrusaitis, J. Shen, D. Chen, F. Wen, Q. Chen, et al. Rodin: A generative model for sculpting 3d digital avatars using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4563–4573, 2023. 1, 3, 4

[55] Z. Wang, C. Lu, Y. Wang, F. Bao, C. Li, H. Su, and J. Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *arXiv preprint arXiv:2305.16213*, 2023. 1, 3

[56] J. Xu, X. Wang, W. Cheng, Y.-P. Cao, Y. Shan, X. Qie, and S. Gao. Dream3d: Zero-shot text-to-3d synthesis using 3d shape prior and text-to-image diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20908–20918, 2023. 3

[57] Y. Xu, H. Tan, F. Luan, S. Bi, P. Wang, J. Li, Z. Shi, K. Sunkavalli, G. Wetzstein, Z. Xu, et al. Dmv3d: Denoising multi-view diffusion using 3d large reconstruction model. *arXiv preprint arXiv:2311.09217*, 2023. 1, 2

[58] W. Yu, L. Yuan, Y.-P. Cao, X. Gao, X. Li, L. Quan, Y. Shan, and Y. Tian. Hifi-123: Towards high-fidelity one image to 3d content generation. *arXiv preprint arXiv:2310.06744*, 2023. 3

[59] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. 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. 4

[60] X.-Y. Zheng, H. Pan, P.-S. Wang, X. Tong, Y. Liu, and H.-Y. Shum. Locally attentional sdf diffusion for controllable 3d shape generation. *ACM Trans. Graph.*, 42(4), jul 2023. 3

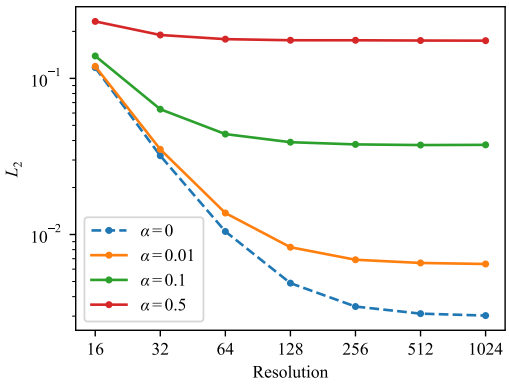


Figure 7. Patch L_2 perturbation of noised images at timestep $t = 200$. $\alpha = 0$ refers to *i.i.d.* noise. As image resolution increases, the L_2 distortion with our proposed noise is almost unaffected and remains at a high level.

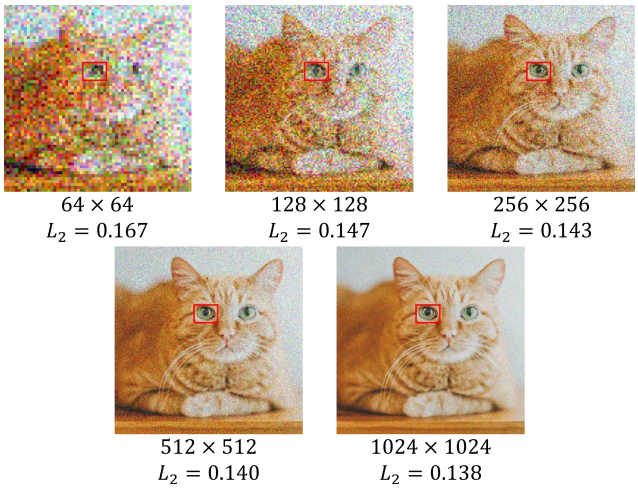


Figure 8. Noised images with different resolutions. All images are noised with $x_t = \sqrt{\gamma}x_0 + \sqrt{1-\gamma}\epsilon$ and $\gamma = 0.65$, $\epsilon \sim \mathcal{N}(0,1)$.

Appendix

A. Low-Frequency Noise

A.1. Derivation of Equation 3 and Equation 5

In this section, we present a detailed derivation of the expected mean L_2 perturbation of a patch in Section 3.2.

Consider a patch $\mathbf{x}_0 = \{x_0^1, x_0^2, \ldots, x_0^M\}$. We add noise $\{\epsilon^i\}_{i=1}^M$ to each value by $x_t^i = \sqrt{\gamma_t}x_0^i + \sqrt{1-\gamma_t}\epsilon^i$ to obtain the noised sample, where γ_t indicates the noise level at timestep t. The expected mean L_2 perturbation of the patch $\mathbf{x}_0$ with *i.i.d.* Gaussian noise $\{\epsilon^i\}_{i=1}^M \sim \mathcal{N}(0,1)$ is

$$
\begin{aligned}
&\mathbb{E}\left(\frac{1}{M}\sum_{i=0}^{M}\left(x_0^i - x_t^i\right)\right)^2 \\
&= \frac{1}{M^2}\mathbb{E}\left(\sum_{i=0}^{M}\left((1-\sqrt{\gamma_t})x_0^i - \sqrt{1-\gamma_t}\epsilon^i\right)\right)^2 \\
&= \frac{(1-\sqrt{\gamma_t})^2}{M^2}\mathbb{E}\left(\sum_{i=0}^{M}x_0^i\right)^2 + \frac{1-\gamma_t}{M^2}\mathbb{E}\left(\sum_{i=0}^{M}\epsilon^i\right)^2 \\
&\quad - \frac{(1-\sqrt{\gamma_t})\sqrt{1-\gamma_t}}{M^2}\mathbb{E}\left(\sum_{\substack{i=0\\j=0}}^{M}x_0^i\epsilon^j\right) \\
&= \frac{(1-\sqrt{\gamma_t})^2}{M^2}\left(\mathbb{E}\sum_{i=0}^{M}\left(x_0^i\right)^2 + \mathbb{E}\sum_{i\neq j}^{M}\left(x_0^i x_0^j\right)\right) \\
&\quad + \frac{1-\gamma_t}{M^2}\left(\mathbb{E}\sum_{i=0}^{M}\left(\epsilon^i\right)^2 + \mathbb{E}\sum_{i\neq j}^{M}\left(\epsilon^i\epsilon^j\right)\right) \\
&= \frac{1}{M}(1-\sqrt{\gamma_t})^2 + \frac{1}{M}(1-\gamma_t) \\
&= \frac{2}{M}\left(1-\sqrt{\gamma_t}\right).
\end{aligned}
\tag{6}
$$

With the proposed low-frequency noise strategy, we mix the noise by $\epsilon^i = \sqrt{1-\alpha}\,\epsilon_1^i + \sqrt{\alpha}\,\epsilon_2$ (Equation 4), where $\{\epsilon_1^i\}_{i=1}^M \sim \mathcal{N}(0,1)$ is independently sampled for each location and $\epsilon_2 \sim \mathcal{N}(0,1)$ is shared within the patch. We still add noise by $x_t^i = \sqrt{\gamma_t}x_0^i + \sqrt{1-\gamma_t}\epsilon^i$ and only ϵ^i is changed. So we have

$$
\begin{aligned}
&\mathbb{E}\sum_{i=0}^{M}\left(\epsilon^i\right)^2 \\
&= \mathbb{E}\sum_{i=0}^{M}\left((1-\alpha)(\epsilon_1^i)^2 + \alpha(\epsilon_2)^2 + 2\sqrt{\alpha(1-\alpha)}\epsilon_1^i\epsilon_2\right) \\
&= (1-\alpha)\mathbb{E}\sum_{i=0}^{M}(\epsilon_1^i)^2 + \alpha\mathbb{E}\sum_{i=0}^{M}(\epsilon_2)^2 \\
&= M,
\end{aligned}
\tag{7}
$$

$$
\begin{aligned}
&\mathbb{E}\sum_{i\neq j}^{M}\left(\epsilon^i\epsilon^j\right) \\
&= \mathbb{E}\sum_{i\neq j}^{M}\left((\sqrt{1-\alpha}\epsilon_1^i + \sqrt{\alpha}\epsilon_2)(\sqrt{1-\alpha}\epsilon_1^j + \sqrt{\alpha}\epsilon_2)\right) \\
&= \mathbb{E}\sum_{i\neq j}^{M}\left((1-\alpha)\epsilon_1^i\epsilon_1^j + \alpha(\epsilon_2)^2\right) \\
&= \alpha\sum_{i\neq j}^{M}\mathbb{E}(\epsilon_2)^2 \\
&= \alpha M(M-1).
\end{aligned}
\tag{8}
$$

In conclusion, the expected mean L_2 perturbation of the patch $\mathbf{x}_0$ with the low-freqency noise is

$$
\begin{aligned}
&\mathbb{E}\left(\frac{1}{M}\sum_{i=0}^{M}\left(x_0^i - x_t^i\right)\right)^2 \\
&= \frac{1}{M}(1-\sqrt{\gamma_t})^2 + \frac{1-\gamma_t}{M^2}\left(M + \alpha M(M-1)\right) \\
&= \frac{2}{M}\left(1-\sqrt{\gamma_t}\right) + \left(1-\frac{1}{M}\right)(1-\gamma_t)\alpha.
\end{aligned} \tag{9}
$$

Here we assume that $\{x_0^i\}_{i=1}^M$ are also sampled from Gaussian distribution $\mathcal{N}(0,1)$, which may be not true on real data. Thus we report the mean L_2 perturbations on real images with different resolutions in Figure 7 as a further demonstration. As illustrated, the L_2 perturbation of *i.i.d.* noise decays exponentially as resolution increases, while our proposed low-frequency noise is slightly affected and converges to larger values proportional to α.

A.2. Justification for patchwise mean L_2 loss

To validate the reasonableness of our adoption of patchwise mean L_2 perturbation, we follow [7] and present an intuitive example using 2D images in Figure 8. The red rectangle highlights the same portion of the object across different resolutions, and we calculate the patchwise L_2 loss for each. We observe that as the image resolution increases, the loss diminishes even though these images maintain the same noise level ($\gamma = 0.65$), making the denoising task easier for networks. Consequently, we believe it is essential to reassess noises from the local patch perspectives and propose the expected mean L_2 perturbation of a patch as a metric.

A.3. Can adjusting the noise schedule also resolve the issue in Figure 3?

In relation to the issue of incomplete removal information in Figure 3 of the main paper, we rely on the low-frequency noise schedule to solve it. However, the question arises: can this issue also be addressed solely by adjusting the noise schedule as mentioned in Section 3.2.2?

The answer is negative. Let's consider a scenario where we modify the noise schedules γ_t and γ_t' for spaces with resolution M and M' respectively, ensuring that the L_2 perturbation remains constant:

$$
\begin{aligned}
&\frac{2}{M'}\left(1-\sqrt{\gamma_t'}\right) = \frac{2}{M}\left(1-\sqrt{\gamma_t}\right) \\
\Leftrightarrow\ &\frac{1-\sqrt{\gamma_t'}}{1-\sqrt{\gamma_t}} = \frac{M'}{M}
\end{aligned} \tag{10}
$$

We take the default setting in Stable Diffusion, where $\beta_T = 0.012$ as an example, leading to $\gamma_T = 0.048$. The volumn resolution (where $M' = 32^3$) is 8 times larger than default resolution ($M = 64^2$). Substituting these values into Equation 10, we find that there is no solution for γ_t'. This suggests that adjusting noise schedule alone is not a viable solution for high-dimensional spaces.

A.4. Forward process

In Section 3.2.3, we propose a novel process from x_0 to x_t with low-frequency noise

$$
\begin{aligned}
x_t^i &= \sqrt{\bar{\alpha}_t}x_0^i + \sqrt{1-\bar{\alpha}_t}\epsilon^i \\
&= \sqrt{\bar{\alpha}_t}x_0^i + \sqrt{(1-\bar{\alpha}_t)(1-\alpha)}\epsilon_1^i + \sqrt{(1-\bar{\alpha}_t)\alpha}\epsilon_2,
\end{aligned} \tag{11}
$$

where $\{\epsilon_1^i\}_{i=1}^M \sim \mathcal{N}(0,1)$ is independently sampled for each location and $\epsilon_2 \sim \mathcal{N}(0,1)$ is shared within the patch. This leads to the conditional density $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{\Sigma})$, where $\mathbf{\Sigma} = \sqrt{1-\alpha}\mathbf{I} + \sqrt{\alpha}\mathbf{J}$.

Alternatively, we can define the one-step transition from x_{t-1} to x_t with low-frequency noise as

$$
\begin{aligned}
x_t^i &= \sqrt{\alpha_t}x_{t-1}^i + \sqrt{1-\alpha_t}\epsilon^i \\
&= \sqrt{\alpha_t}x_{t-1}^i + \sqrt{(1-\alpha_t)(1-\alpha)}\epsilon_1^i + \sqrt{(1-\alpha_t)\alpha}\epsilon_2,
\end{aligned} \tag{12}
$$

which leads to $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)\mathbf{\Sigma})$. We find the definitions in Equation 11 and Equation 12 are essentially equivalent, as we can iterate Equation 12 and derive

$$
\begin{aligned}
&x_t^i = \sqrt{\alpha_t}x_{t-1}^i + \sqrt{(1-\alpha_t)}\epsilon_{t-1}^i \\
&= \sqrt{\alpha_t}x_{t-1}^i + \sqrt{(1-\alpha_t)(1-\alpha)}\epsilon_{1,t-1}^i + \sqrt{(1-\alpha_t)\alpha}\epsilon_{2,t-1} \\
&= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2}^i + \sqrt{(1-\alpha_{t-1})(1-\alpha)}\epsilon_{1,t-2}^i\right. \\
&\quad \left.+\sqrt{(1-\alpha_{t-1})\alpha}\epsilon_{2,t-2}\right) + \sqrt{(1-\alpha_t)(1-\alpha)}\epsilon_{1,t-1}^i \\
&\quad + \sqrt{(1-\alpha_t)\alpha}\epsilon_{2,t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2}^i \\
&\quad + \sqrt{1-\alpha}\left(\sqrt{1-\alpha_t}\epsilon_{1,t-1}^i + \sqrt{\alpha_t(1-\alpha_{t-1})}\epsilon_{1,t-2}^i\right) \\
&\quad + \sqrt{\alpha}\left(\sqrt{1-\alpha_t}\epsilon_{2,t-1} + \sqrt{\alpha_t(1-\alpha_{t-1})}\epsilon_{2,t-2}\right) \\
&= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2}^i + \sqrt{(1-\alpha_t\alpha_{t-1})(1-\alpha)}\tilde{\epsilon}_{1,t-2}^i \\
&\quad + \sqrt{(1-\alpha_t\alpha_{t-1})\alpha}\tilde{\epsilon}_{2,t-2} \\
&= \cdots \\
&= \sqrt{\bar{\alpha}_t}x_0^i + \sqrt{(1-\bar{\alpha}_t)(1-\alpha)}\tilde{\epsilon}_{1,0}^i + \sqrt{(1-\bar{\alpha}_t)\alpha}\tilde{\epsilon}_{2,0}
\end{aligned} \tag{13}
$$

where $\epsilon_1 \sim \mathcal{N}(0,1)$ and $\tilde{\epsilon}_1 \sim \mathcal{N}(0,1)$ are independently sampled for each location and $\epsilon_2 \sim \mathcal{N}(0,1)$ and $\tilde{\epsilon}_2 \sim \mathcal{N}(0,1)$ is shared within the patch.

A.5. Posterior estimation

[17] proposed a training loss for diffusion models

$$\mathbb{E}_q\Bigg[D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\|p(\mathbf{x}_T)) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) + \sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\|p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))\Bigg], \tag{14}$$

where q and p are the forward and reverse process of diffusion models, and provided an estimation of the posteriors

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\mathbf{x}_0 + \sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\mathbf{x}_t}{1-\bar{\alpha}_t}, \frac{(1-\bar{\alpha}_{t-1})(1-\alpha_t)}{1-\bar{\alpha}_t}\mathbf{I}\right) \tag{15}$$

based on $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)\mathbf{I})$ to be isotropic Gaussian. Here we prove that the estimation still holds with the proposed low-frequency noise

$$\begin{aligned}
&q\left(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0\right) \\
&= \frac{q\left(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}\right) q\left(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0\right)}{q\left(\boldsymbol{x}_t|\boldsymbol{x}_0\right)} \\
&= \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\,\boldsymbol{\Sigma}\right) \\
&\quad \cdot \mathcal{N}\left(\boldsymbol{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0, (1-\bar{\alpha}_{t-1})\,\boldsymbol{\Sigma}\right) \\
&\quad \cdot \frac{1}{\mathcal{N}\left(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\,\boldsymbol{\Sigma}\right)} \\
&\propto \exp\left\{-\frac{1}{2}\left[\frac{\left(\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_{t-1}\right)^\top \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_{t-1}\right)}{1-\alpha_t}\right.\right. \\
&\quad + \frac{\left(\boldsymbol{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0\right)^\top \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0\right)}{1-\bar{\alpha}_{t-1}} \\
&\quad \left.\left. - \frac{\left(\boldsymbol{x}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0\right)^\top \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0\right)}{1-\bar{\alpha}_t}\right]\right\} \\
&= \exp\left\{-\frac{1}{2}\left[\left(\frac{\alpha_t}{1-\alpha_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)\boldsymbol{x}_{t-1}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x}_{t-1}\right.\right. \\
&\quad \left.\left. -2\left(\frac{\sqrt{\alpha_t}\boldsymbol{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0}{1-\bar{\alpha}_{t-1}}\right)^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x}_{t-1} + C\left(\boldsymbol{x}_t,\boldsymbol{x}_0\right)\right]\right\} \\
&= \exp\left\{-\frac{1}{2}\left(\boldsymbol{x}_{t-1} - \frac{\sqrt{\bar{\alpha}_{t-1}}\left(1-\alpha_t\right)}{1-\bar{\alpha}_t}\boldsymbol{x}_0\right.\right. \\
&\quad \left.\left. + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)\boldsymbol{x}_t}{1-\bar{\alpha}_t}\right)^\top \left(\frac{\left(1-\bar{\alpha}_{t-1}\right)\left(1-\alpha_t\right)}{1-\bar{\alpha}_t}\boldsymbol{\Sigma}\right)^{-1} \\
&\quad \left(\boldsymbol{x}_{t-1} - \frac{\sqrt{\bar{\alpha}_{t-1}}\left(1-\alpha_t\right)\boldsymbol{x}_0 + \sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)\boldsymbol{x}_t}{1-\bar{\alpha}_t}\right)\Bigg\} \\
&\propto \mathcal{N}\left(\boldsymbol{x}_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\left(1-\alpha_t\right)\boldsymbol{x}_0 + \sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)\boldsymbol{x}_t}{1-\bar{\alpha}_t}, \right. \\
&\quad \left. \frac{\left(1-\bar{\alpha}_{t-1}\right)\left(1-\alpha_t\right)}{1-\bar{\alpha}_t}\boldsymbol{\Sigma}\right),
\end{aligned} \tag{16}$$

exp	N	C	PSNR ↑	SSIM ↑	LPIPS ↓
#1	32	32	27.83	0.874	0.276
#2	32	4	27.69	0.874	0.279
#3	64	4	29.21	0.886	0.228
#4	$32 \rightarrow 64$	4	28.68	0.883	0.167

Table 5. Ablation experiments on resolution N and channel C of the volume encoder.

where $C\left(\boldsymbol{x}_t,\boldsymbol{x}_0\right)$ is constant up to $\boldsymbol{x}_t,\boldsymbol{x}_0$, $\boldsymbol{\Sigma} = \sqrt{1-\alpha}\mathbf{I} + \sqrt{\alpha}\mathbf{J}$, $\mathbf{I}$ is identity matrix and $\mathbf{J}$ is all-one matrix. Therefore, we can train diffusion models with the proposed low-frequency noise.

B. Volume Encoder

In Table 5, we conducted ablation experiments to study how resolution N, channel C and loss term affects the performance of the volume encoder.

We find the channel C of volume is a minor factor of the performance of the volume encoder. In contrast, increasing the resolution N greatly improves the reconstruction performance. However, $N = 64$ brings a computation and GPU memory cost that is 8 times larger than $N = 32$, which causes significant difficulty for training diffusion models.

In order to increase volume resolution without large overhead, we introduce a super-resolution module before we feed the generated volume into the refinement module. We increase the spatial resolution of the volume from $N = 32$ to $N = 64$. The super-resolution module composed of few layers of 3D convolution is served as a post-process and is performed on the outputs of the diffusion model. In our experiments, the super-resolution approach achieves close performances comparing to native $N = 64$ volumes. The diffusion model is trained on the volumes with lower resolution $N = 32$, and the rendering is performed on the upsampled volumes with higher resolution $N = 64$. Therefore, we can enjoy both a lower dimension for easier training of diffusion models as well as a higher resolution for rendering more detailed textures without much overhead.

C. More results

We kindly present more detailed illustrations in the supplementary files and video. We render more results generated with our method in Figure 9 and Figure 10. We emphasize the diversity in Figure 11 and the flexibility in Figure 12 of our method. Also, we provide more comparisons with other text-to-3D approaches in Figure 13.

Figure 9. More text-to-3D generations of VolumeDiffusion.

Figure 10. More text-to-3D generations of VolumeDiffusion.

Figure 11. Diverse text-to-3D generations of VolumeDiffusion.

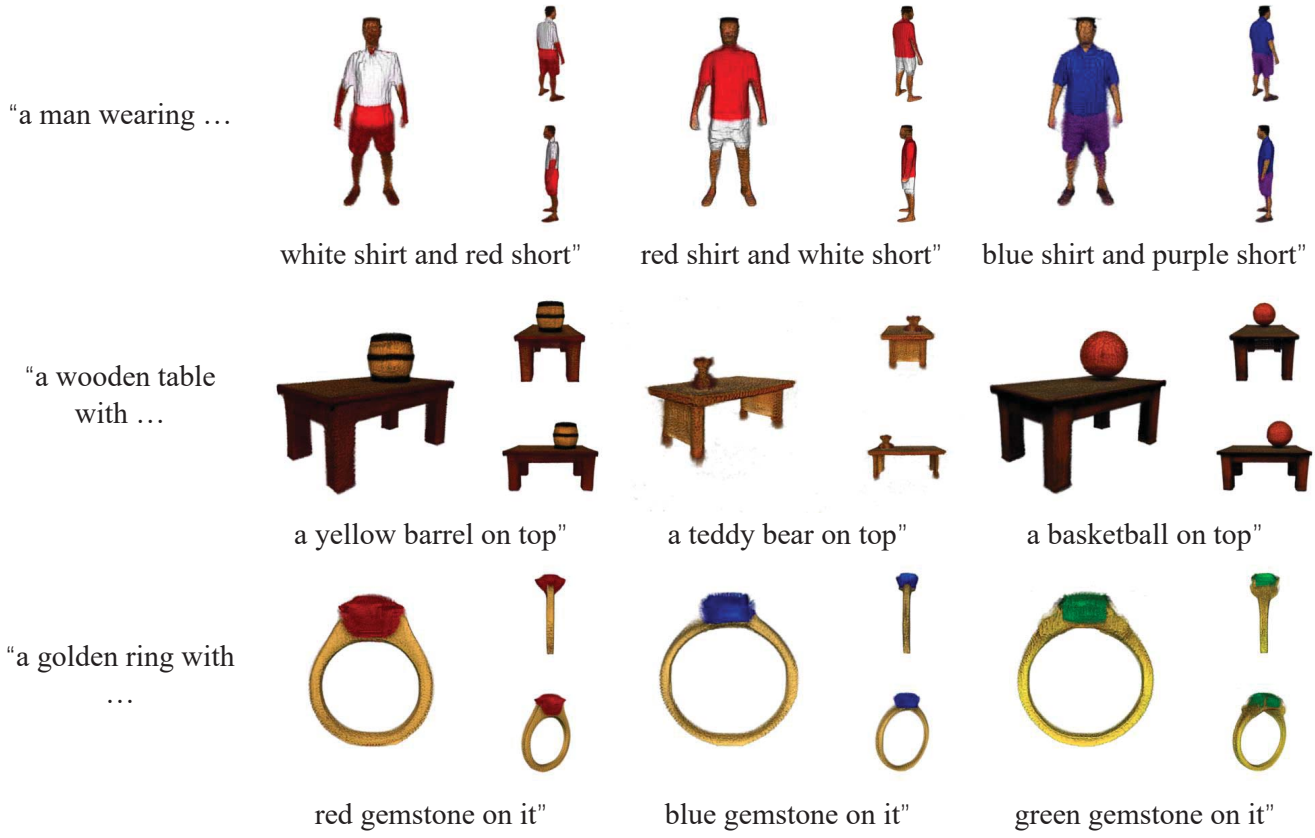


Figure 12. Flexible text-to-3D generations of VolumeDiffusion.

Figure 13. Comparison with other text-to-3D methods.