High-Fidelity 3D Textured Shapes Generation by Sparse Encoding and Adversarial Decoding

Qi Zuo[1*], Xiaodong Gu[1*], Yuan Dong[1*], Zhengyi Zhao[1], Weihao Yuan[1], Lingteng Qiu[2], Liefeng Bo[1], and Zilong Dong[1†]

[1] Institute of Intelligent Computing, Alibaba Group
[2] SSE, CUHKSZ

Fig. 1: Exported textured shapes of our single-class unconditional generation models. All models are exported using an UV unwrapper *xatlas* [39] and rendered using Mitsuba [16].

Abstract. 3D vision is inherently characterized by sparse spatial structures, which propels the necessity for an efficient paradigm tailored to 3D generation. Another discrepancy is the amount of training data, which undeniably affects generalization if we only use limited 3D data. To solve these, we design a 3D generation framework that maintains most of the building blocks of StableDiffusion with minimal adaptations for textured shape generation. We design a Sparse Encoding Module for details preservation and an Adversarial Decoding Module for better shape recovery. Moreover, we clean up data and build a benchmark on the biggest 3D dataset (**Objaverse**). We drop the concept of 'specific class' and treat the 3D Textured Shapes Generation as an open-vocabulary problem. We

* Equal contribution.
† Corresponding author: list.dzl@alibaba-inc.com.

first validate our network design on **ShapeNetV2** with 55K samples on single-class unconditional generation and multi-class conditional generation tasks. Then we report metrics on processed **G-Objaverse** with 200K samples on the image conditional generation task. Extensive experiments demonstrate our proposal outperforms SOTA methods and takes a further step towards open-vocabulary 3D generation. We release the processed data at https://aigc3d.github.io/gobjaverse/.

Keywords: 3D Generation · Sparse Encoding · Adversarial Decoding

1 Introduction

The vision community has witnessed the great success of 2D artificial intelligence-generated content (AIGC) methods such as StableDiffusion [36] and GigaGAN [18]. This encourages us to rethink what we need to enable 3D generation.

When adding another dimension to the 2D generation task, the space complexity increases from O^2 to O^3, leading to significant computation growth. Therefore, 3D assets are more difficult to fit into the neural networks than 2D assets. Sparsity is a distinct attribute of 3D vision that has been utilized in large-scale scene perception tasks like 3D detection and semantic segmentation of point clouds but merely discussed in 3D generation. Our key motivation in this paper, however, is that sparsity is vital for 3D generation, especially when we need to generate high-frequency textures simultaneously.

Signed Distance Function (SDF) [7,8,29], and Occupancy [44,55,59] are typical dense representations in 3D generation. Although they have advantages in accurate shape modeling, their shortcomings are memory costing and inaccurate alignment to texture, which limits their performance in textured shape generation. Methods using point clouds as representation are naturally sparse, yet they lose detailed structure information. To recover the structural information, conversions from sparse space to dense space and then reversely from dense space to sparse space are used for point cloud applications and demonstrate the highest quality of generated shapes [25, 49, 60]. To avoid the laborious conversion, Sparse Convolution Networks [10, 47] are proposed to directly process sparse point clouds, which demonstrate state-of-the-art (SOTA) performance on large-scale vision perception tasks [15, 27]. Hence in this work, we propose to use a sparse encoding module to alleviate the high computation complexity in 3D shape compression. Since we use a sparse encoding module to extract per-coordinate features, these features can be further projected to 2D planes by predefined angles. Then these 2D feature planes can be compressed by general 2D encoders.

A key difference with the geometry generation task is that we need to simultaneously generate the high-frequency texture attached to the underlying low-frequency geometry. Unlike previous geometry generation methods [7, 8, 29, 44, 55, 59] which use thousands of points or a low-resolution feature volume as feed-forward representation, texture generation requires the high-frequency surface color features to be maintained as much as possible. Although the sparse

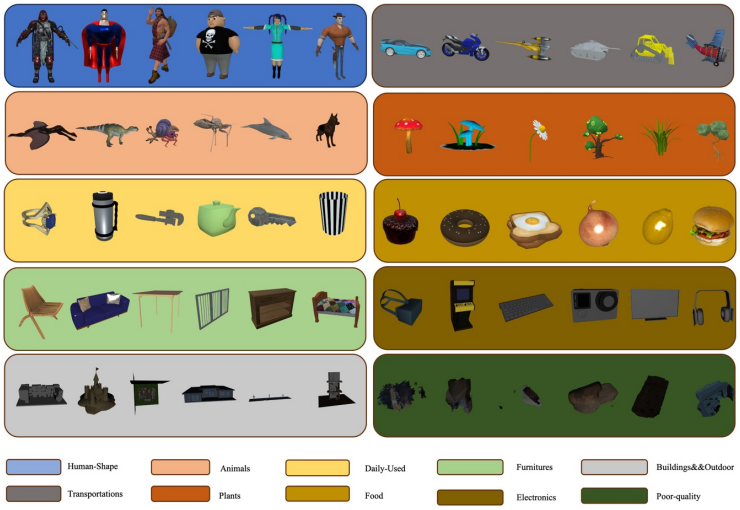

Fig. 2: We manually split Objaverse into 10 general classes as the color bands depict. Note we do not use the *Building&&Outdoor* and *PoorQuality* classes, since we empirically find that they are harmful to model convergence and we put further analysis in the appendix. By splitting Objaverse into general classes, we can build a benchmark on it, which we have achieved by shuffling class data and splitting it into certain proportions respectively for training, validation, and testing.

encoding module can preserve most of the information, the regularization item (KL-divergence) on the latent space will block the feature flow to a certain extent. To recover the lost information, we introduce a Dual MarchingCube layer [41] on the decoding stage which can render and extract textured shapes differentiably and we further conduct extra adversarial training [36] on the renderings to enhance the texture quality. We adopt the naive StableDiffusion framework which sets up a diffusion model in the latent space for generation purpose and greatly reduces the computation resources needed. Compared with StableDiffusion, the differences are the conversion from 3D dense point clouds to 2D feature planes in the encoding stage and the conversion from 2D feature planes to 3D textured shapes in the decoding stage. With minimal modifications and enhancements, our framework can generate a batch of high-fidelity textured shapes in less than 5 seconds using a DDIM [46] sampler.

We formulate our design theoretically and experimentally validate it on the public benchmark ShapeNetV2 [4] and further scale up it to Objaverse [11]. During our experiment, we clean the Objaverse dataset and manually label 10 general classes which gives about 200K samples as depicted in Fig. 2. We split 8 general classes into train/val/test splits and built a benchmark for the evaluation of open-vocabulary image-conditional generation. We conduct three kinds of experiments to demonstrate the effectiveness of our method including single-class un-

conditional generation, multi-class conditional generation, and open-vocabulary image-conditional generation. Experiments on both datasets demonstrate the fidelity and variety of our generated samples. We claim our main contributions include:

1) We analyze and propose a novel sparse encoding and dense decoding paradigm (Sparse3D) that can achieve both high-fidelity single-class generation and open-vocabulary 3D textured shapes generation.

2) We manually filter out and split the Objaverse dataset into 10 general classes (200K shapes) and construct a benchmark (G-Objaverse) for open-vocabulary 3D textured shapes generation.

3) We demonstrate SOTA performance on both datasets and we conduct extensive experiments to confirm the insight of design. We also further give an outlook on future works.

2 Related Work

Geometry Generation Many works in the 3D vision community seek to solve problems with geometry first. Also in the generation task, pioneer works mainly focus on the geometry structure, where various representations are applied. Methods which adopt point cloud [53,54,60], occupancy [28,33,40,44], and signed distance function (SDF) [7,9,12,59] are mainly geometry-only. Although they are short of generating texture for the underlying representation, their generations can potentially maintain the quality of man-made shapes, which is of vital importance for subsequent manipulation like texturing or editing. A potential improvement in the geometry-only generation is the adaptation to large-scale 3D datasets like Objaverse [11], which currently may be hindered by the uneven quality of its 3D assets.

Texture Generation As a complementary part of works on geometry generation, texture generation aims to generate feasible texture on fixed geometry given text prompts. From rendering-based [45], inpainting-based [5] to optimization-based [1, 58] methods, they are faced with the challenge of keeping multi-view consistency, which we believe can be relieved by introducing explicit 3D inductive bias (Objaverse). Approaches like a multi-view consistent depth inpainting ControlNet or a 3D-conditioned (voxel or SDF) ControlNet can potentially improve the consistency of texture generation. However, all these potential improvements require a large amount of clean, high-quality 3D assets, which again motivates us to select high-quality 3D assets from Objaverse.

3D-Aware Generative Models As a twin task, novel view synthesis (NVS) focuses on the quality of multi-view renderings rather than the quality of underlying geometry and texture. Previous works [57, 61] using 2D GANs for NVS often fall short in multi-view consistency. 3D-aware methods [2, 3, 30, 32] thus adopt neural radiance fields or multi-level surfaces as their internal implicit representations, directly introducing 3D prior constraints and achieving impressive

view synthesis quality on both object-level and scene-level. Recently we have witnessed diffusion-based NVS methods [62] like Zero123 [23], SyncDreamer [24], Consistent123 [52], Wonder3D [26] and Zero123++ [42]. Although some of them try to inject 3D-priori like coarse volume or multi-view correlations, they do not directly enforce an underlying 3D representation like previous GAN-based 3D-aware methods, which is a potential future work. A common shortcoming of NVS methods is that they need to use MarchingCubes for explicit mesh extraction first and then query the field to obtain vertex color and the best threshold for each shape needs to be dynamically adjusted, which may make the final geometry shattered.

Lifting 2D into 3D Due to the lack of 3D assets and the strong capability of 2D generation models, some works also try to lift 2D images by rendering poses into 3D representation. DreamFusion [34] first combines neural fields with a fixed Imagen [38] and it uses score distillation loss to update the neural field according to specific prompts. Follow-up works [6, 20, 21, 35, 43, 48, 51] mainly focus on elevating the resolution of 3D representation, optimizing the stability under various text prompts, avoiding over-saturated textures, accelerating and dealing with the Janus problem. Though those methods achieve semantic control and high fidelity under some prompts, they fall short in the long optimization time and instability with random text prompts.

Textured Shapes Generation Recently, methods [13, 63] that can simultaneously output raw mesh and corresponding texture (*i.e.* Textured Shapes) come out in a 3D-supervised way. Compared with NVS methods, these methods produce more feasible topologies like man-made shapes and can also generate corresponding textures. GET3D [13] is the first method capable of generating explicit free-topology textured shapes in an end-to-end manner. It incorporates a hybrid tri-plane representation together with a differentiable marching tetrahedron layer into a StyleGAN2 for end-to-end single-class textured shape generation. Since massive 3D contents collected from websites are hard to align, the camera pose-conditioned paradigm of GET3D can not be generalized to multi-class and open-vocabulary scenarios. Another work that should be noticed and most similar to our approach is 3DGen [14], which utilizes a triplane-based Variational AutoEncoder(VAE) to convert the dense point clouds to tetrahedron meshes. Also, a triplane stands as the latent representation and is fed through a diffusion UNet [37] for generation purposes. It utilizes Objaverse for pretraining and then finetunes the model on ShapeNetV2, achieving well generalizability on multi-class conditional generation. However, the limitations of 3DGen lie in the design of the autoencoding stage. It also uses the UNet structure for encoding and decoding, which keeps the resolution of the triplane but hinders the optimization of the latent diffusion model. In our design, we show that there is no need to keep the semantic layout of projected triplanes, and we can treat the feature planes like 2D natural images in the StableDiffusion pipeline.

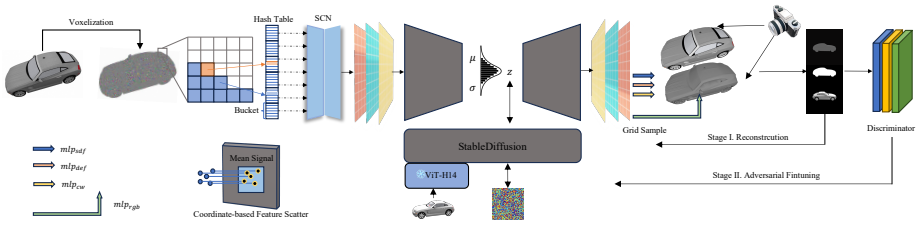

Fig. 3: Sparse3D only varies with StableDiffusion in several specific components. In the input stage, a dense point cloud (**with 1M to 4M points**) is voxelized by a resolution of 1000^3, and then fed to a Sparse Convolutional Network to extract coordinate-based features. To align the huge amount of sparse features to a dense representation, we project multiple features onto specific pixel grids and compute the mean values of these features. After we translate 3D dense point clouds into 2D feature maps, we finetune the pre-trained StableDiffusion for 2D feature map generation. In the meanwhile, we decode the feature maps as explicit mesh through a differentiable mesh extraction layer (Flexicubes) and then optimize the variational autoencoder using a rendering-based reconstruction penalty. Since our output contains RGB renderings in a similar domain with natural images, we further use an N-layer discriminator for adversarial finetuning to enhance the texture quality after reconstruction converges.

3 Method

The framework of our method is illustrated in Fig. 3, which comprises three parts: Sparse Encoding, Generative Differentiable Mesh Decoding, and Latent Diffusion Module. In this section, we first present preliminaries in Sec. 3.1, after which the specific design of Sparse Encoding is stated in Sec. 3.2 and Adversarial Differentiable Mesh Decoding is stated in Sec. 3.3. Further, we introduce the LDM design of our case in Sec. 3.4.

3.1 Preliminaries

Sparse Convolution. In geometry generation, a common input consists of thousands of points, which is sufficient for geometry generation but not for texture generation. We need more input information to recover the surface texture. Traditional 3D convolutional neural networks use a unit grid as a dense representation which often leads to explosive memory growth when we increase the resolution of the unit grid. As an alternative, in large-scale scene understanding which often formulates the input as point clouds with several million points, sparse convolution is adopted to extract the structure features. For a point cloud with point coordinates $p \in \mathbf{R}^{N\times 3}$ and point features $p \in \mathbf{R}^{N\times C_f}$, sparse convolution uses sparse structures like octree [50] or hashtable [10, 47] to store these information and network operators are only applied on sparse points. To prove the necessity of using sparse convolution, we show the sparse quantization results

in Supp-Fig. 1, which shows that it is useless if we only increase the number of input points with a low-resolution voxelization.

Differentiable Mesh Extraction. Conversion from raw meshes to implicit fields like occupancy, signed distance function, and NeRF usually introduces computation errors, while differentiable mesh extraction demonstrates an elegant way to optimize the underlying implicit field in a probabilistic manner. In our framework, we adopt the most recent Flexicubes [41] representation as it is more elaborately designed to recover the structure details.

3.2 Sparse Encoding

For a long time, 3D representation has been fed to the network as a unit cube with much free space useless. Simply increasing the resolution of the unit cube is effective but burdens much on computation and memory. Sparsity is a unique characteristic of the real 3D world which is different from dense projections (images). Starting from sparsity, we design a Sparse Encoding Module for encoding 3D assets into latent space. The input of this module is a dense colored point cloud $p \in \mathbf{R}^{N\times 6}$ with a random number of points N. We do not decline the points to a certain number like previous approaches since it is important for recovering high-frequency information (texture). We then feed the batched dense point clouds to a Sparse Convolutional Network (SCN) and obtain point-wise features. For simplicity, we directly adopt a UNet-Style SCN from MinkowaskiiEngine [10]. The feature is further projected to the triplane space $f_{in} \in \mathbf{R}^{3C_f\times K_f\times K_f}$ through coordinate-based feature scatter. Then we directly apply the encoder of StableDiffusion E_{sd} to f_{in} after modifying the input projection layer. The output of modified encoder $E_{sd}^{'}$ is the concatenation of mean μ and log variance $ln(\sigma^2)$ in a gaussian distribution. Then we employ the KL-Divergence for regularization:

$$\begin{aligned} &\mathcal{C}(\mu, ln(\sigma^2)) = E_{sd}^{'}(\mathcal{P}(\mathcal{S}(p))), \\ &L_{kl} = -\frac{1}{2b}\sum(1 + ln(\sigma^2) - \mu^2 - \sigma^2), \end{aligned} \tag{1}$$

where $\mathcal{S}$ denotes the sparse feature extraction, $\mathcal{P}$ denotes the feature projection based on coordinates, $\mathcal{C}$ denotes concatenation, and b denotes the batch size. The usage of the SCN for feature extraction is the key to maintaining dense low-level features of the high-frequency texture.

3.3 Adversarial Differentiable Decoding

Given μ and $ln(\sigma^2)$, we apply the reparameterization trick to obtain a resampled latent code z, and we also reuse the decoder structure D_{sd} of StableDiffusion. We modify the output projection layer of D_{sd} and denote the new non-linear transformation as $D_{sd}^{'}$. The output of $D_{sd}^{'}$ is the concatenation of geometry feature

planes $f_{geo} \in \mathcal{R}^{3C_{geo} \times K_f \times K_f}$ and texture feature planes $f_{tex} \in \mathcal{R}^{3C_{tex} \times K_f \times K_f}$, sharing the same resolution K_f as the input feature planes.

We denote the geometry query positions of Flexicubes as $Q^{fl}_{geo} \in \mathcal{R}^{N^{fl}_{geo} \times 3}$ and use grid sample to query the corresponding feature in f_{geo}, resulting in $f^{Q}_{geo} \in \mathcal{R}^{N^{fl}_{geo} \times C_{geo}}$. In Flexicubes, the input of the differentiable dual marching cube layer includes the SDF, deformation, and cube weights at Q^{fl}_{geo}. We thus define three MLPs for them, respectively. The above process can be written as:

$$\begin{aligned} f_{sdf} &= mlp_{sdf}(gs(f_{geo}, Q^{fl}_{geo})) \in \mathcal{R}^{N^{fl}_{geo} \times 1}, \\ f_{def} &= mlp_{def}(gs(f_{geo}, Q^{fl}_{geo})) \in \mathcal{R}^{N^{fl}_{geo} \times 3}, \\ f_{cw} &= mlp_{cw}(gs(f_{geo}, Q^{fl}_{geo})) \in \mathcal{R}^{N^{fl}_{geo} \times 21}, \end{aligned} \tag{2}$$

where gs denotes grid sample. The f_{cw} in Eq. (2) has a feature dimension of 21, which is composed of $\beta \in \mathcal{R}^{N^{fl}_{geo} \times 12}$, $\alpha \in \mathcal{R}^{N^{fl}_{geo} \times 8}$, and $\gamma \in \mathcal{R}^{N^{fl}_{geo} \times 1}$. β, α is used to adjust dual vertices positioning, and γ is used to control the splitting of quadrilaterals into triangles. The final vertices and faces can be extracted from f_{sdf}, f_{def}, and f_{cw} in a differentiable manner. We refer the readers to Flexicubes [41] for more details about the process.

Since we already have explicit vertices and faces, surface rendering techniques [19] can be used to obtain the depth map I_d and alpha I_a under a certain camera pose ξ. Moreover, the rasterization process will generate query positions for texture $Q^{fl}_{tex} \in \mathcal{R}^{N^{fl}_{tex} \times 3}$ under a certain rendering resolution, which can be used to query f_{tex} and be further decoded into RGB colors f_{rgb} by an MLP:

$$f_{rgb} = mlp_{rgb}(gs(f_{tex}, Q^{fl}_{tex})) \in \mathcal{R}^{N^{fl}_{tex} \times 3}, \tag{3}$$

where RGB map I_c can be obtained by clamping f_{rgb} in $[0, 1]$ and projecting the clamped RGB values back to pixel space. Now, we can easily define the differentiable rendering-based reconstruction penalty as:

$$L_{re} = \lambda_{depth} L_{depth} + \lambda_{alpha} L_{alpha} + \lambda_{color} L_{color}, \tag{4}$$

where $L_{depth} = |I_d - I^{gt}_d|_1$, $L_{alpha} = |I_a - I^{gt}_a|_1$, and $L_{color} = |I_c - I^{gt}_c|_1$. gt denotes the ground truth renderings and $|\cdot|_1$ denotes absolute difference.

To further improve the texture quality, we borrow a simple N-layer discriminator from StableDiffusion and formulate an adversarial paradigm. To avoid collapses, we only conduct adversarial training when the pure reconstruction converges. The vanilla generative penalty can be written as:

$$\begin{aligned} L_G =& \mathbb{E}_{z \in \mathcal{N}(\mu, \sigma^2)}[-\mathcal{D}(I_c)], \\ L_D =& \mathbb{E}_{z \in \mathcal{N}(\mu, \sigma^2), I^{gt}_c}[-\mathcal{D}(I^{gt}_c) + \mathcal{D}(I_c)], \end{aligned} \tag{5}$$

where $\mathcal{D}$ denotes the N-layer discriminator. In all, we first train the encoder and decoder using the following penalty,

$$L_{NoAdv} = \lambda_{kl} L_{kl} + L_{re}. \tag{6}$$

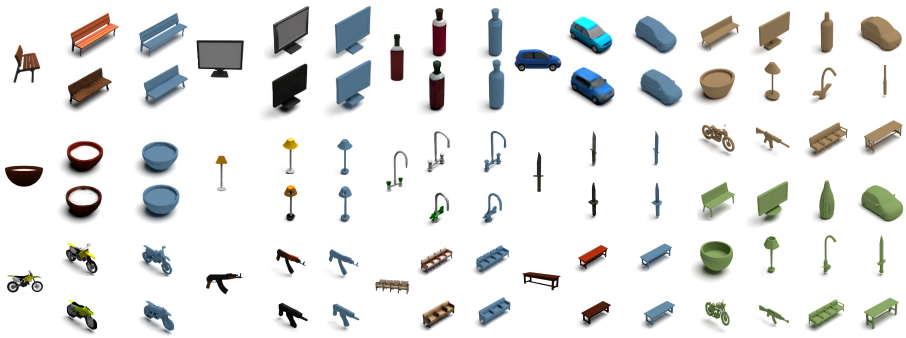

Fig. 4: Qualitative conditional generation results on ShapeNetV2. For each conditioned image, the result of our method stands in the first line and the result of TexturedLAS stands in the second line. The last four columns show the shapes that are queried from **3DILG** and **3DS2Vec** using OpenShape [22].

After the reconstruction converges, we conduct adversarial training using the following objective.

$$L_{Adv} = \lambda_{kl} L_{kl} + L_{re} + \lambda_{adv} L_G \tag{7}$$

3.4 Latent Diffusion Model

We use off-the-shell 2D Diffusion Model [7] as our base model and load the pre-trained weight when the training process starts. To enable large-scale conditional generation, we use a vision transformer trained on laion2b for image feature extraction, and we use cross-attention for condition injection at different scales. The vision transformer is frozen during the training process since it tends to overfit a specific concept when trained on a small dataset. Note that we do not implement a text encoder since we find current text captions of publicly available 3D models are of poor quality, and we hold the concept that text-to-3d should leverage the 2D generation models to formulate a text-to-image-to-3D paradigm which is more controllable by introducing a powerful text-to-image generation model.

4 Experiments

4.1 Datasets

We split our experiments into three folds: 1) Single-class unconditional generation comparison on three classes of ShapeNetV2 (Car, Chair, and Table) following the train/valid split in previous methods [13]; 2) Multi-class conditional generation comparison on all classes of ShapeNetV2, also following the

train/valid split in previous methods [55]; 3) Label-free conditional generation comparison on the 8 general classes from Objaverse which is labeled manually, and the splits are generated by randomly shuffling and splitting the data into proportions of $0.9, 0.05, 0.05$ respectively for train, validation, and test.

4.2 Hyperparameters and Details

In Sec. 3, our description involves many hyperparameters and hidden details which should be clarified. We render each 3D model into 38 views of 1024×1024 resolution including depth, alpha, and RGB images with blender. The rendering poses are illustrated in the appendix in detail and we refer readers to the appendix for more information. Since different shapes have different ratios of occupancy of the rendering images, the reprojected point clouds may have different amounts of points N, which range mostly from 1 million to 4 million. The point feature dimension C_f is set to 12, and the plane resolution K_f is set to 256. The shape of the latent code of our VAE is the same as StableDiffusion (*i.e.* $[4, 32, 32]$), and we set the weight of KL-Divergence λ_{kl} to $1e^{-7}$. For simplicity, we also put the structure of our SCN in the appendix and we refer the readers to the appendix for more technique details. In the decoding stage, we set $C_{geo} = 8$, $C_{tex} = 8$, and we use Flexicubes with a resolution of 96 for all experiments. In the final loss, we set $\lambda_{depth} = 2$, $\lambda_{alpha} = 2$, $\lambda_{color} = 1$, and $\lambda_{adv} = 0.01$.

Table 1: Quantitative Metrics on ShapeNetV2 classes *Car*, *Chair* and *Table*.

Category	Method	COV-CD↑(%)	MMD-CD↓($\times 10^{-3}$)	FID-3D↓
Car	GET3D [13]	44.00	2.28	10.54
	Sparse3D	**77.71**	**1.24**	**10.17**
Chair	GET3D [13]	76.63	5.81	33.18
	Sparse3D	**83.28**	**3.33**	**18.85**
Table	GET3D [13]	73.4	5.73	26.05
	Sparse3D	**82.90**	**3.72**	**11.44**

4.3 Single-Class Unconditional Generation

Quantitative Comparison. We conduct the quantitative comparison to the recent single-class unconditional generation model, GET3D [13]. For a fair comparison, we also render 26 views according to the script provided by GET3D and set the object scale to 0.9, which means that the mesh will be normalized in $[-0.45, 0.45]$. We keep the same setting in both our method and GET3D. FID and Chamfer Distance are reported in Tab. 1. Note we find GET3D needs to carefully choose a feasible normalized scale and weight of regularization for a

Table 2: Quantitative Metrics of multi-class autoencoding on ShapeNetV2.

Metric	Category	3DILG	3DS2Vec	Sparse3D	
				Surface	Structure
Chamfer↓	table	0.026	0.026	**0.019**	**0.023**
	car	0.066	0.062	**0.025**	**0.050**
	chair	0.029	**0.027**	**0.023**	0.030
	airplane	0.019	**0.017**	**0.014**	**0.017**
	sofa	0.030	**0.029**	**0.018**	0.038
	rifle	0.017	**0.014**	**0.011**	**0.014**
	lamp	0.036	**0.032**	**0.026**	0.037
	mean(selected)	0.032	**0.030**	**0.019**	**0.030**
	mean(all)	0.040	0.038	**0.026**	**0.037**

Table 3: Quantitative comparison on ShapeNetV2(**S**) between TexturedLAS and Sparse3D and metrics achieved by Sparse3D on Objaverse datasets(**O**).

Method	Chamfer Dist.↓	↑ PSNR↑	SSIM↑	LPIPS↓
TexturedLAS(***S***)	0.149	18.79	0.901	0.121
Sparse3D(***S***)	**0.081**	**20.03**	**0.921**	**0.099**
Sparse3D(***O***)	0.108	19.23	0.908	0.110

specific class, and the training of class *Table* collapses at scale 0.9. We further render the data of *Table* at scale 0.7 and report related metrics instead. For the computation of Chamfer Distance, we normalize both the generated shapes and ground truth shapes of the test dataset in a unit space and report Coverage (COV) and Minimum Matching Distance (MMD) evaluation metrics of two groups of shapes.

Qualitative Comparison. Since it is hard to align two unconditional methods for visualization, for qualitative comparison, we only generate and export textured shapes for each class using our method. As depicted in Fig. 1, Sparse3D faithfully generates high-quality samples in the single-class unconditional generation on various classes. We further show the interpolation result of two random samplings in Supp-Fig. 6, which demonstrates the smoothness of our distribution mapping function.

4.4 Multi-Class Conditional Generation

Quantitative Comparison. As textured shape generation on multi-class conditional generation of ShapeNetV2 does not have a public benchmark currently,

we conduct experiments with geometry-only SOTA methods, 3DILG [55] and 3DS2Vec [56], and further compare it with TexturedLAS[‡], which is based on LASDiffusion [59] but can simultaneously output occupancy and a texture field. We refer the readers to the appendix for our implementation details of TexturedLAS, which is also an early technical exploration of us. For 3DILG and 3DS2Vec, we only report the autoencoding-related metrics since they have not released image-conditioned pre-trained weights. For TexturedLAS, we report geometry-related metrics (Chamfer Distance) and rendering-related metrics (PSNR, SSIM, and LPIPS) for image-conditional generation. As depicted in Tab. 2, we directly adopt the metrics reported in 3DILG and 3DS2Vec and use the dataset splits released by 3DS2Vec. We normalize each generated shape in a unit cube between $[-1, 1]$ and follow the same evaluation script with 3DS2Vec. Since we train our framework by differentiable rendering, we report chamfer distance in **Surface** and **Structure** settings. For the Structure setting, we sample points in all faces of a mesh and compute the chamfer distance between two point clouds, which is the same as 3DS2Vec. For the Surface setting, we sample points from the rendering surface of a mesh, which means that we do not consider the internal structure. As a rendering-based method, we should only report the Surface setting but we find that in the Structure setting, Sparse3D is comparable to the SOTA shape generation method 3DS2Vec and sometimes even slightly better, which demonstrates the superiority of our design.

In textured shapes generation, we report metrics on the whole test dataset of ShapeNetV2. The final metrics are the average of 55 classes and the metrics of each class are the average metrics of each instance. As depicted in Tab. 3, Sparse3D achieves better geometry and texture generation quality than TexturedLAS for the better ability to preserve high-frequency features. Although we elevate the resolution of the occupancy diffusion stage from 64 to 128, TexturedLAS still performs poorly both on geometry and texture generation.

Qualitative Comparison. We visualize conditional generation results of baselines and our method in Fig. 4. As is shown, our method can generate smooth topology without direct 3D supervision and generate texture seamlessly with the underlying geometry. TexturedLAS can also generate smooth and high-quality geometry but can not generate decent texture since it is based on volume representation. Note we have increased the resolution of the first stage in TexturedLAS to 128, but we can not eliminate the black surface phenomenon which we further explain in the appendix. We use shapes generated by our conditional generation as a reference and use OpenShape [22] to search for similar shapes in the sampling results of 3DS2Vec and 3DILG. The final visualization also shows that the quality of the underlying geometry of generated shapes of Sparse3D is comparable to or even slightly better than methods that directly use 3D supervision (SDF).

[‡] Verification code will be released at https://github.com/hitsz-zuoqi/TexturedLAS

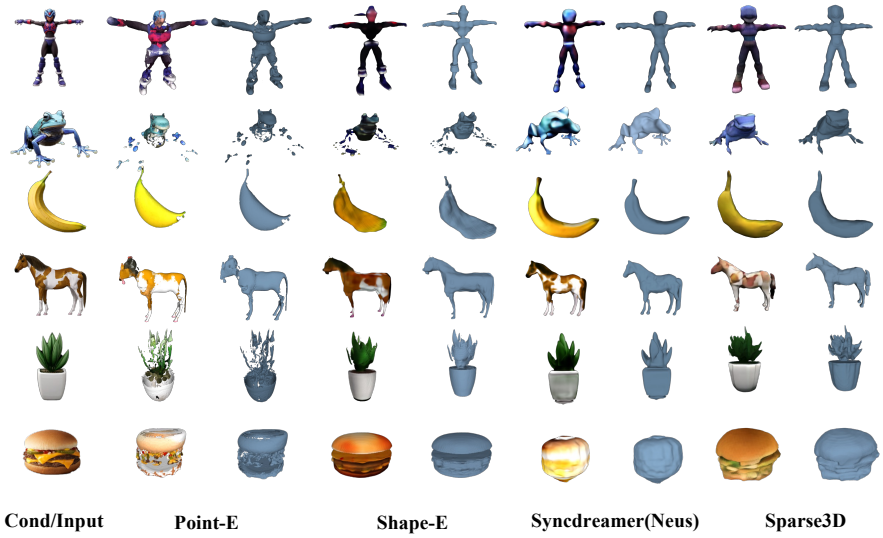

Fig. 5: Qualitative results on various image-to-3D methods. Due to different settings, the image is treated as either a condition or the input.

4.5 Label-free Image-Conditional Generation

Quantitative Results. Since previous methods are not trained on the same datasets as Sparse3D, we only report our metrics on the testing dataset as a baseline for future work. As also depicted in Tab. 3, Sparse3D achieves poorer metrics on Objaverse than ShapeNetV2. The reasons behind this are two-fold: 1) Objaverse contains various materials that will set obstacles for us to obtain the ground-truth renderings using open-source rendering engines like Blender; 2) Objaverse contains more complicated samples than CAD models in ShapeNetV2. We leave these potential improvements to future work.

Qualitative Comparison. We visualize some typical samples generated by Sparse3D, Point-E [31], Shape-E [17], and Syncdreamer(Neus) [24]. Point-E and Shap-E are trained on private datasets with several million 3D models. Syncdreamer is trained on the whole Objaverse. Since all of these methods use different training datasets, we only aim to illustrate typical strengths and problems among current methods. As shown in Fig. 5, Sparse3D could generate topologies like man-made shapes but fails to generate accurate texture based on the conditioned image. We empirically find that an unaligned dataset(Objaverse) will cause ambiguity in image-3D alignment since two objects with the same topology but different rotations are treated as distinct objects in networks. Point-E and Shap-E suffer from the same ambiguity problem as Sparse3D both in geometry and texture. SyncDreamer may generate a corrupted bottom for the

generated fixed up-hemisphere views. Current methods are hard to benchmark for the reason of using different training sets, which again motivates us to build up and release the benchmark. However, we show competitive visualization results in the open-vocabulary scenario. Also, as shown in Supp-Fig. 7, Sparse3D can be adopted for a text-to-image-to-3D pipeline rather than directly learning a mapping from text prompts to 3D models.

4.6 Ablation

We conduct an ablation study on significant modifications including sparse encoding and adversarial fintuning. As depicted in Tab. 4, sparse encoding greatly improves both the geometry-related and texture-related metric since it preserves more features of original shapes. Adversarial finetuning further enhances the texture quality of reconstructed shapes, which builds a higher upbound for the generation model. (The term "**baseline**" indicates that we only used a 20K point cloud as input for PointNet, as opposed to a dense point cloud in the order of millions as input for SparseConv.)

Table 4: Ablation on Sparse Encoding and Adversarial Fintuning.

Method	IoU	CD	PSNR	SSIM
Baseline	0.951	0.039	25.83	0.946
+SparseEncoding	0.954	**0.036**	26.11	0.951
+Adv Fintuning	0.952	0.039	27.03	0.954
Full Model	**0.954**	0.037	**27.59**	**0.959**

5 Conclusion

We propose Sparse3D, a novel framework that utilizes an off-the-shell 2D generation model (StableDiffusion) for 3D textured shape generation. Minimal modifications stand in the sparse encoding stage, which is aimed to boost the quality of texture and the adversarial finetune as a second stage of reconstruction. Experiments on the single-class unconditional setting, multi-class conditional setting, and open-vocabulary conditional setting demonstrate the superiority of Sparse3D. Although we take a step further on textured shape generation on large-scale scenarios and produce shapes similar to man-made ones, the data limitation of 3D generation has not been fully resolved yet and the generalizability of Sparse3D is relatively weak when compared with NVS methods. In future works, we want to combine the generalizability of NVS methods with the topology-preserving ability of Sparse3D and incorporate an explicit 3D prior to the NVS methods to formulate a 3D-aware diffusion model.

References

1. Cao, T., Kreis, K., Fidler, S., Sharp, N., Yin, K.: Texfusion: Synthesizing 3d textures with text-guided image diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4169–4181 (2023)
2. Chan, E.R., Lin, C.Z., Chan, M.A., Nagano, K., Pan, B., De Mello, S., Gallo, O., Guibas, L.J., Tremblay, J., Khamis, S., et al.: Efficient geometry-aware 3d generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16123–16133 (2022)
3. Chan, E.R., Monteiro, M., Kellnhofer, P., Wu, J., Wetzstein, G.: pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5799–5809 (2021)
4. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository. Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago (2015)
5. Chen, D.Z., Siddiqui, Y., Lee, H.Y., Tulyakov, S., Nießner, M.: Text2tex: Text-driven texture synthesis via diffusion models. arXiv preprint arXiv:2303.11396 (2023)
6. Chen, R., Chen, Y., Jiao, N., Jia, K.: Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. arXiv preprint arXiv:2303.13873 (2023)
7. Cheng, Y.C., Lee, H.Y., Tulyakov, S., Schwing, A.G., Gui, L.Y.: Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4456–4465 (2023)
8. Chou, G., Bahat, Y., Heide, F.: Diffusion-sdf: Conditional generative modeling of signed distance functions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2262–2272 (2023)
9. Chou, G., Chugunov, I., Heide, F.: Gensdf: Two-stage learning of generalizable signed distance functions. Advances in Neural Information Processing Systems **35**, 24905–24919 (2022)
10. Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 3075–3084 (2019)
11. Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., Farhadi, A.: Objaverse: A universe of annotated 3d objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13142–13153 (2023)
12. Dong, Y., Zuo, Q., Gu, X., Yuan, W., Zhao, Z., Dong, Z., Bo, L., Huang, Q.: Gpld3d: Latent diffusion of 3d shape generative models by enforcing geometric and physical priors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 56–66 (June 2024)
13. Gao, J., Shen, T., Wang, Z., Chen, W., Yin, K., Li, D., Litany, O., Gojcic, Z., Fidler, S.: Get3d: A generative model of high quality 3d textured shapes learned from images. Advances In Neural Information Processing Systems **35**, 31841–31854 (2022)

14. Gupta, A., Xiong, W., Nie, Y., Jones, I., Oğuz, B.: 3dgen: Triplane latent diffusion for textured mesh generation. arXiv preprint arXiv:2303.05371 (2023)
15. Gwak, J., Choy, C., Savarese, S.: Generative sparse detection networks for 3d single-shot object detection. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16. pp. 297–313. Springer (2020)
16. Jakob, W., Speierer, S., Roussel, N., Vicini, D.: Dr.jit: A just-in-time compiler for differentiable rendering. Transactions on Graphics (Proceedings of SIGGRAPH) **41**(4) (Jul 2022). https://doi.org/10.1145/3528223.3530099
17. Jun, H., Nichol, A.: Shap-e: Generating conditional 3d implicit functions. arXiv preprint arXiv:2305.02463 (2023)
18. Kang, M., Zhu, J.Y., Zhang, R., Park, J., Shechtman, E., Paris, S., Park, T.: Scaling up gans for text-to-image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10124–10134 (2023)
19. Laine, S., Hellsten, J., Karras, T., Seol, Y., Lehtinen, J., Aila, T.: Modular primitives for high-performance differentiable rendering. ACM Transactions on Graphics **39**(6) (2020)
20. Li, W., Chen, R., Chen, X., Tan, P.: Sweetdreamer: Aligning geometric priors in 2d diffusion for consistent text-to-3d. arXiv preprint arXiv:2310.02596 (2023)
21. Lin, C.H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.Y., Lin, T.Y.: Magic3d: High-resolution text-to-3d content creation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 300–309 (2023)
22. Liu, M., Shi, R., Kuang, K., Zhu, Y., Li, X., Han, S., Cai, H., Porikli, F., Su, H.: Openshape: Scaling up 3d shape representation towards open-world understanding. arXiv preprint arXiv:2305.10764 (2023)
23. Liu, R., Wu, R., Van Hoorick, B., Tokmakov, P., Zakharov, S., Vondrick, C.: Zero-1-to-3: Zero-shot one image to 3d object. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9298–9309 (2023)
24. Liu, Y., Lin, C., Zeng, Z., Long, X., Liu, L., Komura, T., Wang, W.: Syncdreamer: Generating multiview-consistent images from a single-view image. arXiv preprint arXiv:2309.03453 (2023)
25. Liu, Z., Tang, H., Lin, Y., Han, S.: Point-voxel cnn for efficient 3d deep learning. Advances in Neural Information Processing Systems **32** (2019)
26. Long, X., Guo, Y.C., Lin, C., Liu, Y., Dou, Z., Liu, L., Ma, Y., Zhang, S.H., Habermann, M., Theobalt, C., et al.: Wonder3d: Single image to 3d using cross-domain diffusion. arXiv preprint arXiv:2310.15008 (2023)
27. Mao, J., Xue, Y., Niu, M., Bai, H., Feng, J., Liang, X., Xu, H., Xu, C.: Voxel transformer for 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3164–3173 (2021)
28. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4460–4470 (2019)
29. Mittal, P., Cheng, Y.C., Singh, M., Tulsiani, S.: Autosdf: Shape priors for 3d completion, reconstruction and generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 306–315 (2022)
30. Nguyen-Phuoc, T., Li, C., Theis, L., Richardt, C., Yang, Y.L.: Hologan: Unsupervised learning of 3d representations from natural images. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7588–7597 (2019)

31. Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., Chen, M.: Point-e: A system for generating 3d point clouds from complex prompts. arXiv preprint arXiv:2212.08751 (2022)
32. Niemeyer, M., Geiger, A.: Giraffe: Representing scenes as compositional generative neural feature fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11453–11464 (2021)
33. Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A.: Convolutional occupancy networks. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16. pp. 523–540. Springer (2020)
34. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. arXiv preprint arXiv:2209.14988 (2022)
35. Qiu, L., Chen, G., Gu, X., Zuo, Q., Xu, M., Wu, Y., Yuan, W., Dong, Z., Bo, L., Han, X.: Richdreamer: A generalizable normal-depth diffusion model for detail richness in text-to-3d (2023), `https://arxiv.org/abs/2311.16918`
36. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10684–10695 (2022)
37. Ronneberger, O., Fischer, P., Brox, T.: 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. pp. 234–241. Springer (2015)
38. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E.L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al.: Photorealistic text-to-image diffusion models with deep language understanding. Advances in Neural Information Processing Systems **35**, 36479–36494 (2022)
39. Sander, P.V., Snyder, J., Gortler, S.J., Hoppe, H.: Texture mapping progressive meshes. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. pp. 409–416 (2001)
40. Sanghi, A., Chu, H., Lambourne, J.G., Wang, Y., Cheng, C.Y., Fumero, M., Malekshan, K.R.: Clip-forge: Towards zero-shot text-to-shape generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18603–18613 (2022)
41. Shen, T., Munkberg, J., Hasselgren, J., Yin, K., Wang, Z., Chen, W., Gojcic, Z., Fidler, S., Sharp, N., Gao, J.: Flexible isosurface extraction for gradient-based mesh optimization. ACM Transactions on Graphics (TOG) **42**(4), 1–16 (2023)
42. Shi, R., Chen, H., Zhang, Z., Liu, M., Xu, C., Wei, X., Chen, L., Zeng, C., Su, H.: Zero123++: a single image to consistent multi-view diffusion base model. arXiv preprint arXiv:2310.15110 (2023)
43. Shi, Y., Wang, P., Ye, J., Long, M., Li, K., Yang, X.: Mvdream: Multi-view diffusion for 3d generation. arXiv preprint arXiv:2308.16512 (2023)
44. Shue, J.R., Chan, E.R., Po, R., Ankner, Z., Wu, J., Wetzstein, G.: 3d neural field generation using triplane diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20875–20886 (2023)
45. Siddiqui, Y., Thies, J., Ma, F., Shan, Q., Nießner, M., Dai, A.: Texturify: Generating textures on 3d shape surfaces. In: European Conference on Computer Vision. pp. 72–88. Springer (2022)
46. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502 (2020)

47. Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., Han, S.: Searching efficient 3d architectures with sparse point-voxel convolution. In: European conference on computer vision. pp. 685–702. Springer (2020)
48. Tang, J., Ren, J., Zhou, H., Liu, Z., Zeng, G.: Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. arXiv preprint arXiv:2309.16653 (2023)
49. Vahdat, A., Williams, F., Gojcic, Z., Litany, O., Fidler, S., Kreis, K., et al.: Lion: Latent point diffusion models for 3d shape generation. Advances in Neural Information Processing Systems **35**, 10021–10039 (2022)
50. Wang, P.S., Liu, Y., Guo, Y.X., Sun, C.Y., Tong, X.: O-cnn: Octree-based convolutional neural networks for 3d shape analysis. ACM Transactions On Graphics (TOG) **36**(4), 1–11 (2017)
51. Wang, Z., Lu, C., Wang, Y., Bao, F., Li, C., Su, H., Zhu, J.: Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. arXiv preprint arXiv:2305.16213 (2023)
52. Weng, H., Yang, T., Wang, J., Li, Y., Zhang, T., Chen, C., Zhang, L.: Consistent123: Improve consistency for one image to 3d object synthesis. arXiv preprint arXiv:2310.08092 (2023)
53. Yang, G., Huang, X., Hao, Z., Liu, M.Y., Belongie, S., Hariharan, B.: Pointflow: 3d point cloud generation with continuous normalizing flows. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 4541–4550 (2019)
54. Zeng, X., Vahdat, A., Williams, F., Gojcic, Z., Litany, O., Fidler, S., Kreis, K.: Lion: Latent point diffusion models for 3d shape generation. arXiv preprint arXiv:2210.06978 (2022)
55. Zhang, B., Nießner, M., Wonka, P.: 3dilg: Irregular latent grids for 3d generative modeling. Advances in Neural Information Processing Systems **35**, 21871–21885 (2022)
56. Zhang, B., Tang, J., Niessner, M., Wonka, P.: 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models (2023), `https://arxiv.org/abs/2301.11445`
57. Zhang, P., Zhang, B., Chen, D., Yuan, L., Wen, F.: Cross-domain correspondence learning for exemplar-based image translation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5143–5153 (2020)
58. Zhao, Z., Song, C., Gu, X., Dong, Y., Zuo, Q., Yuan, W., Dong, Z., Bo, L., Huang, Q.: An optimization framework to enforce multi-view consistency for texturing 3d meshes using pre-trained text-to-image models (2024), `https://arxiv.org/abs/2403.15559`
59. Zheng, X.Y., Pan, H., Wang, P.S., Tong, X., Liu, Y., Shum, H.Y.: Locally attentional sdf diffusion for controllable 3d shape generation. arXiv preprint arXiv:2305.04461 (2023)
60. Zhou, L., Du, Y., Wu, J.: 3d shape generation and completion through point-voxel diffusion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5826–5835 (2021)
61. Zhou, X., Zhang, B., Zhang, T., Zhang, P., Bao, J., Chen, D., Zhang, Z., Wen, F.: Cocosnet v2: Full-resolution correspondence learning for image translation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11465–11475 (2021)
62. Zuo, Q., Gu, X., Qiu, L., Dong, Y., Zhao, Z., Yuan, W., Peng, R., Zhu, S., Dong, Z., Bo, L., Huang, Q.: Videomv: Consistent multi-view generation based on large video generative model (2024), `https://arxiv.org/abs/2403.12010`

63. Zuo, Q., Song, Y., Li, J., Liu, L., Bo, L.: Dg3d: Generating high quality 3d textured shapes by learning to discriminate multi-modal diffusion-renderings. In: 2023 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 14529–14538 (2023). https://doi.org/10.1109/ICCV51070.2023.01340