TexOct: Generating Textures of 3D Models with Octree-based Diffusion

Jialun Liu, Chenming Wu[1]*, Xinqi Liu, Xing Liu, Jinbo Wu, Haotian Peng,
Chen Zhao, Haocheng Feng, Jingtuo Liu, Errui Ding
Department of Computer Vision Technology(VIS), Baidu Inc.
liujialun@baidu.com

Abstract

This paper focuses on synthesizing high-quality and complete textures directly on the surface of 3D models within 3D space. 2D diffusion-based methods face challenges in generating 2D texture maps due to the infinite possibilities of UV mapping for a given 3D mesh. Utilizing point clouds helps circumvent variations arising from diverse mesh topologies and UV mappings. Nevertheless, achieving dense point clouds to accurately represent texture details poses a challenge due to limited computational resources. To address these challenges, we propose an efficient octree-based diffusion pipeline called TexOct. Our method starts by sampling a point cloud from the surface of a given 3D model, with each point containing texture noise values. We utilize an octree structure to efficiently represent this point cloud. Additionally, we introduce an innovative octree-based diffusion model that leverages the denoising capabilities of the Denoising Diffusion Probabilistic Model (DDPM). This model gradually reduces the texture noise on the octree nodes, resulting in the restoration of fine texture. Experimental results on ShapeNet demonstrate that TexOct effectively generates high-quality 3D textures in both unconditional and text / image-conditional scenarios.

1. Introduction

3D models, represented by polygon meshes, are widely used in visual applications like videos, games, and VR / AR scenarios. The quality of texture maps is vital for the appearance of 3D models across various applications. In previous decades, creating high-quality textures typically required experienced designers to invest substantial effort, resulting in considerable consumption of both time and cost. However, recent advancements in creating content with neural networks have brought new energy and perspective to this area of computer vision and graphics.

A common method for handling texture maps is UV representation, which converts 3D textures to a set of 2D images using UV coordinate transformation. This method effectively links 3D texture generation with 2D image generation techniques, such as GAN [13, 18, 31, 35, 46] and Diffusion models [10, 27, 29, 30]. However, constructing a unified UV mapping for all shapes of 3D objects presents challenges, making it difficult to generate 3D textures from 2D UV space directly. To bypass the challenges, recent studies [4, 5, 8, 32] investigated the application of 2D generation techniques for producing multi-view textures for 3D objects. These methods focus on ensuring consistency across different views, which is crucial for realistic and coherent texturing. However, due to self-occlusion, these methods face limitations in finding an optimal set of views that can adequately cover the entire surface of a 3D object. This becomes an instance of the notorious NP-hard *set cover problem* [11], and maintaining pixel-perfect multi-view consis-

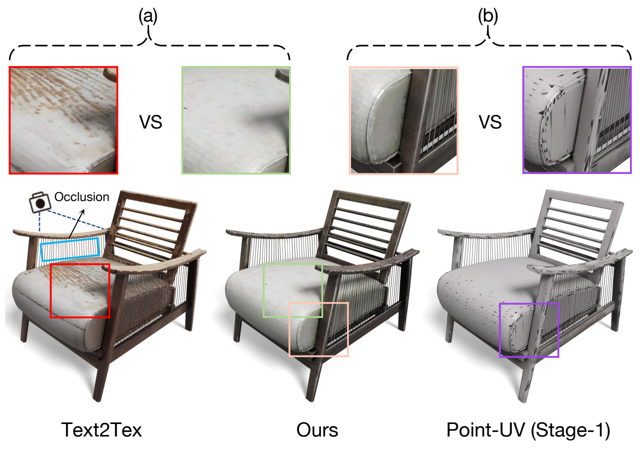

Figure 1. Comparisons with different methods: (a) In Text2Tex [8], self-occlusion (the blue box) results in texture errors (the red box). In contrast, our method of directly generating textures in 3D space avoids issues caused by self-occlusion. (the green box). (b) Point-UV [43] (Stage-1) utilizes limited point cloud data to capture 3D information, resulting in rough textures (the purple box). In comparison, our method generates high-quality details (the orange box) with denser point cloud.

*Corresponding author.

tency is arduous. For example, in Figure 1-(a), the surface texture generated by Text2Tex [8] for the chair is incorrect. This error occurs because the optimization view is obscured by self-occlusion.

To create a comprehensive and consistent texture for a given 3D mesh, one straightforward method is to generate the texture directly on the object's surface in 3D space. Among various techniques, point clouds are widely recognized as an effective method for representing 3D information. The level of detail in textures represented by point clouds is influenced by the density of points sampled on the surface. In essence, the greater the number of points sampled, the more detailed the texture representation on the 3D object becomes. However, owing to constraints of the GPU memory capacity, previous texture generation methods [7, 9, 12, 42, 43] have been limited to synthesizing relatively low-resolution textures with low density of point clouds. As illustrated in Figure 1-(b), Point-UV [43](Stage-1) utilizes only a few points, leading to the generation of rough 3D textures.

In this paper, we introduce a novel octree-based diffusion approach called *TexOct* that directly optimizes texture generation in 3D space. TexOct effectively alleviates the above issues, *i.e.*, 1) texture errors caused by self-occlusion, and 2) texture degradation caused by sparse sampling.

TexOct consists of two key components, *i.e.*, octree construction and octree-based diffusion model learning. Firstly, we utilize the octree structure to represent the dense point cloud sampled from the surface of 3D model. Octree is a highly efficient data structure that organizes the point cloud in a tree-like format. Octree begins with the initial point cloud as the root node and recursively divides it into 8 children nodes, merging points with similar coordinates. The subdivision will continue until we reach a predetermined depth of the octree. Secondly, The constructed octree is fed into TexOct, which aims to synthesize a realistic texture from the noisy texture values assigned to each octree node. Specifically, TexOct is implemented as a U-Net architecture with ResNet blocks. It incorporates octree-based operators developed in OCNN [37] to perform DDPM process on octree structure. Additionally, TexOct also supports conditional texture generation. We develop an octree-based cross-attention module after each ResNet block (except for the first one), allowing for effective text / image information integration into the generation process, thereby enabling texture generation to be guided by text / image conditions.

The main contributions of this paper can be summarized as follows:

- We introduce an end-to-end 3D texture generation approach in 3D space, avoiding texture map errors caused by self-occlusion in multi-view generation approaches.
- We propose an innovative octree-based 3D diffusion method that effectively utilizes the densely sampled point cloud on the 3D model's surface for high-quality texture generation.
- The qualitative and quantitative evaluations demonstrate that our method performs well in unconditional and conditional generation contexts. Additionally, the user study indicates that the texture generated by our method is more favored by users.

2. Related work

Texture Generation. In recent years, several methods [2, 15, 25, 41] have been proposed for modeling and generating textures on 3D shapes. Chen et al. [10] introduced AUV-Net for modeling texture representation on 3D shapes. It embeds 3D surfaces into a 2D-aligned UV space for easy texture synthesis and transfer. The alignment is learned by a network in an unsupervised manner. Siddiqui et al. [32] proposed an end-to-end trainable GAN equipped with novel convolutional and pooling layers, that directly operate on 3D surfaces. Their model learns to synthesize realistic textures for 3D shapes from the 2D image domain, addressing the problem of insufficient data. Cao et al. [5] proposed a 3D texture generation approach utilizing a 2D diffusion model. Given a 3D shape, the method aggregates multi-view latent texture maps at each denoising step using a consistent UV map. Their approach fuses multiple denoising processes to generate geometrically consistent 3D textures without any fine-tuning. Bokhovkin et al. [4] proposed a GAN-based approach for realistic texture generation given 3D shapes. The method learns a texture manifold on mesh faces utilizing densely sampled points and 2D adversarial training loss. The approach generalizes to image-guided 3D texture synthesis. Chen et al. [8] proposed a multi-view projection-based approach for 3D texture synthesis utilizing a 2D diffusion model with text conditioning. The method allows for dynamic mask generation to prevent 3D inconsistencies and artifacts. They also employ a view sequence generation scheme to optimally update the partial texture. Yu et al. [43] propose a two-stage framework involving point diffusion and UV diffusion. In the coarse stage, The method samples a number of points on the mesh surface and generates a coarse texture using the point diffusion model. In the fine stage, a UV diffusion model is used to refine the coarse texture, resulting in a smooth texture image. Richardson et al. [28] proposed a method to texture a 3D shape by iteratively projecting 2D images back onto its surface. They utilized a depth-guided diffusion model for generating these 2D images and introduced an aggregation strategy to minimize artifacts.

3D Model Generation with Textures. Gupta et al. [14] proposed a method for simultaneously generating the mesh and texture of 3D content. This method utilizes a tri-plane Variational Autoencoder (VAE) to learn representations of textured meshes. To enable text-controlled generation, a

2D diffusion model is employed within the latent space of the tri-plane VAE. Wu et al. [39] proposed Sin3DM which learns geometry and texture from a single 3D textured mesh. The method employs a latent diffusion model within the tri-plane latent space. Various applications such as retargeting, outpainting, and local editing are demonstrated. Zheng et al. [45] proposed a two-stage diffusion approach for generating novel shapes controlled by user-defined 2D sketch images. In the first stage, the method learns to predict the occupancy field of the generated shape using the sketch image as guidance. In the second stage, an additional diffusion model is incorporated to super-resolve the occupancy voxels, thereby generating high-resolution 3D shapes.

3D Diffusion. Zeng et al. [44] proposed a latent diffusion model that operates on point clouds. The model includes a VAE and two latent diffusion networks, which construct hierarchical representations of point clouds using the two diffusion networks. Yu et al. [43] focused on 3D texture generation, they proposed to apply diffusion models on a given point cloud to generate texture in a progressive manner. Nakayama et al. [21] studied controllable 3D point cloud generation. They proposed an approach that models the independent parts of objects. Wu et al. [40] focused on controlling the point cloud generation using sketch images and texts. They proposed an approach that first extracts features from an image and a prompt and then generates a point cloud based on the features using a joint diffusion pipeline. Nichol et al. [22] proposed a transformer-based approach for point cloud generation. Their approach employs two diffusion models sequentially for text-to-image and image-to-point cloud conversions. Wu et al. [38] addressed the need for fast point cloud generation. They proposed an approach that optimizes the curvy learning trajectory of point diffusion into a straight path and develops a strategy to shorten the path.

3. Method

In this section, we describe how the proposed approach effectively models accurate textures represented by a dense point cloud. Given a 3D object, we aim to learn a diffusion model for directly generating a realistic and high-quality 3D texture for it in 3D space. To accomplish this, we introduce a novel and efficient diffusion model called TexOct, which performs the denoising process of DDPM on the octree structure. The framework of our method is illustrated in Figure 2. Firstly, we sample a large number of points on the surface of a 3D mesh to capture the detailed structure of the object. To efficiently process the point cloud data, we represent the point cloud using an octree. Secondly, we propose an octree-based diffusion model, which is a U-Net architecture. This model takes the octree as input and gradually denoises the texture noise value of the octree. Next, through a process called "Reverse Octree," we obtain a colored point cloud. Finally, we map the colored points back onto the mesh to generate a textured mesh.

3.1. Preliminary

Before introducing our method, we provide a brief overview of some fundamental concepts necessary for understanding diffusion models (DDPMs) [17, 23]. Gaussian diffusion models assume a forward noising process which gradually applies noise to real data x_0. The forward process is the variance-preserving Markov process [33] specified as $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$, where constants $\bar{\alpha}_t$ are hyperparameters. With the reparameterization trick, we can express x_t as $\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, \mathbf{I})$. For the reverse process, diffusion models are trained to learn a denoising network for inverting the forward process. The reverse process is represented as $p_\theta(x_{t-1}|x_t) = \mathcal{N}(\mu_\theta(x_t), \Sigma_\theta(x_t))$, where neural networks are used to predict the statistics of p_θ. The denoising model is trained with the variational lower bound [19] of the log-likelihood of x_0, which simplifies to:

$$\mathcal{L} = -p_\theta(x_0|x_1) + \sum_t \mathcal{D}_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)) \tag{1}$$

where $\mathcal{D}_{KL}(\cdot||\cdot)$ represents the KL divergence that measure the discrepancy the mean and covariance of two distributions. Since both $q(x_{t-1}|x_t, x_0)$ and $p_\theta(x_{t-1}|x_t)$ are Gaussians, we can predict the noise ϵ_θ by reparameterizing μ_θ. Consequently, the model can be trained using mean-squared error loss between the predicted noise $\epsilon_\theta(x_t)$ and the ground truth sampled Gaussian noise ϵ_t. The training objective can be formulated as

$$\mathcal{L}_{simple} = ||\epsilon_\theta(x_t) - \epsilon_t||_2^2 \tag{2}$$

Once p_θ is trained, new sample can be sampled by initializing $x_{t_{\text{max}}} \sim \mathcal{N}(0, \mathbf{I})$ and sampling $x_{t-1} \sim p_\theta(x_{t-1}|x_t)$ via the reparameterization trick.

3.2. Octree Construction

Octree is a hierarchical data structure used for spatial subdivision in 3D. It provides an efficient approach to organizing a point cloud. In our method, we utilize the mesh-sampling [20] to sample a set of points $\mathcal{P} = \{\mathbf{p}_i\}_{i=1}^M$ from the surface of 3D mesh. A point is defined by x, y, z, r, g, b. Specifically, x, y, z is the coordinates. r, g, b is its ground-truth texture. Given The point cloud $\mathcal{P}$, we translate the point cloud by an *offset* = (min($\mathcal{P}_x$), min($\mathcal{P}_y$), min($\mathcal{P}_z$)), and quantize it by quantization step *qs*

$$\mathcal{P}_Q = round(\frac{\mathcal{P} - offset}{qs}) \tag{3}$$

$$qs \geq \frac{max(P) - min(P)}{2^L - 1} \tag{4}$$

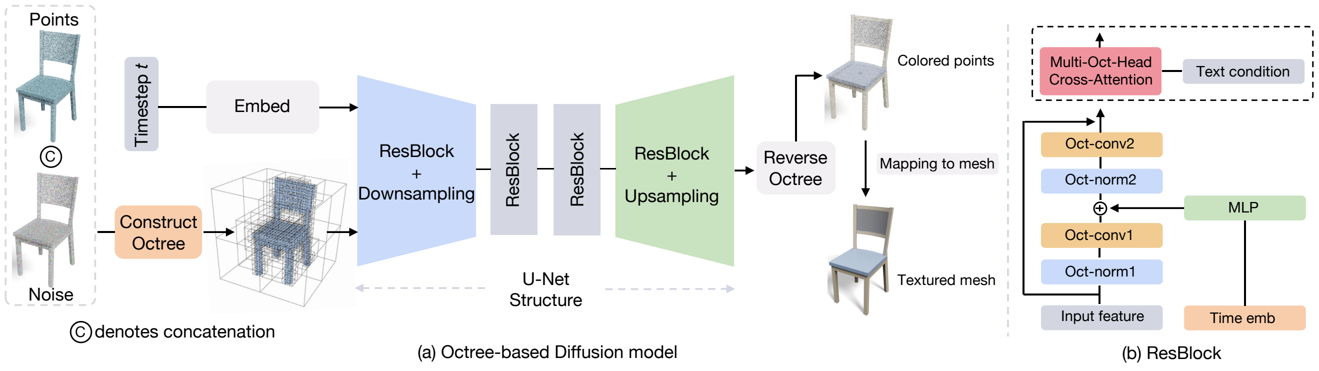

Figure 2. The overview of TexOct. (a) The architecture of the Octree-based diffusion model. We start by sampling a set of points from the surface of a 3D object mesh. The points combined with the noised texture are used to construct the octree. The octree and the timestep t serve as the input for the octree-based diffusion model. The diffusion model reduces the texture noise of the octree to obtain a colored one. The octree can be transformed back into the original points along with the generated color, utilizing the "reverse octree" option. Finally, we map the colored points into a textured mesh. (b) The architecture of ResBlock. Each ResBlock comprises two octree convolutions(Oct-conv), two octree normalizations(Oct-norm), and a MLP. In case of a text-conditional diffusion model, we include a multi-oct-head cross-attention module after each ResBlock.

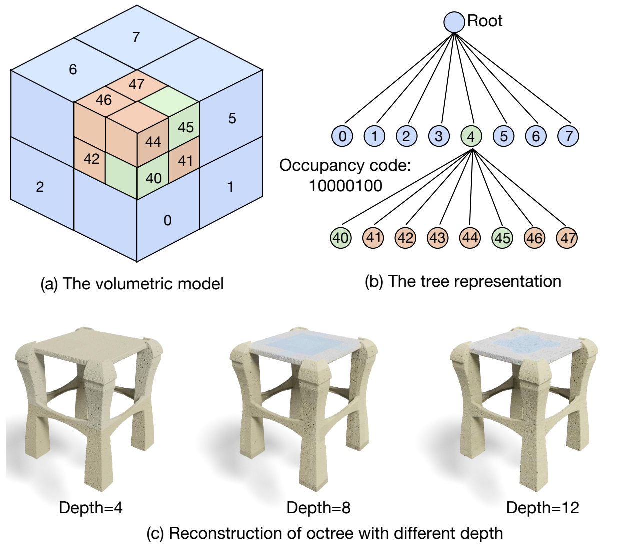

Figure 3. A toy example for constructing the octree. The volumetric model is shown in (a) and the corresponding tree representation on (b). In (c), the point cloud of the same object is quantized by octree with the max depth of $4, 8, 12$.

In Figure 3, we present a toy example to demonstrate the construction procedure of an octree. Figure 3-(a) shows the initial volumetric model, while Figure 3-(b) illustrates the corresponding tree representation of the octree after two subdivision processes.

Initially, the octree divides the cube space based on the maximum side length of the bounding box of $\mathcal{P}_Q$ into 8 equal octants recursively. The occupancy status of 8 children cubes is encoded using an 8-bit binary occupancy node. The occupancy code of the root node is 00010000, indicating that only child node-4 is occupied and will undergo further subdivision, while the other children nodes remain unoccupied. In the second subdivision, node-4 is further divided into 8 children nodes. The occupancy code of node-4 becomes 10000100, indicating that child node-40 and node-45 are occupied and will undergo further subdivision, while the other children nodes remain unoccupied. This subdivision process continues until the maximum depth L is reached, resulting in the complete construction of the octree. At the leaf nodes of the octree, each 8-bit occupancy code represents 8 cubes with a side length equal to the quantization step qs. The points in P are aligned and merged with the nearest corresponding cube.

The depth L determines the resolution of the octree. A higher value of L corresponds to a higher resolution octree. Figure 3-(c) illustrates the reconstruction of the octree for the same 3D object, exhibiting different depths ranging from 4 to 12. As the depth of the octree increases, the resolution of the octree improves. This can be observed through the higher level of detail captured in the reconstruction. Additionally, in accordance with Eq. 3 to Eq. 5, the increase in depth leads to a reduction in the reconstruction error.

To reconstruct the point cloud, we perform inverse quantization to get the reconstructed point cloud $\tilde{P}$. The reconstruction error is controlled in:

$$error = \max_i \left\| \tilde{P}_i - P_i \right\|_\infty \leq \frac{qs}{2} \tag{5}$$

3.3. Octree-based Diffusion Model Learning

Overview. The proposed octree-based diffusion model operates on the nodes of the constructed octrees. It restores the noisy texture values assigned to each node to a version that is likely sampled from a manifold representing natural textural surfaces. We implement our diffusion model using a U-Net architecture with ResNet blocks. This model consists of octree-based operators as developed in [37]. Our U-Net design comprises four stages, where each stage processes the features at a different tree depth. Specifically, the tree depths for these stages are 12, 11, 10, and 9, respectively in our experiment. This enables the model to learn information from different receptive fields along with an octree, thereby allowing the model to generate fine details with global consistency. The overall structure of the proposed network is depicted in Fig. 2-(a). In Fig. 2-(b), we show the architecture of an individual ResNet block. We cascade an octree-based attention module when we employ the block for conditional generation. It is noteworthy that the attention module is omitted from the block in the first stage of the U-Net to enhance memory efficiency. We provide the detailed settings of the proposed model in the supplementary material.

Training. Given a training sample that is a point cloud with texture values, we first convert it to an octree using the method described in 3.2. Then, we randomly sample a noise pad and amplify it with a randomly chosen time step t. We compute the noisy texture for a tree node by applying the amplified noise pad to the texture values assigned to the node as described in 3.1. We repeat this process for all the octree nodes to complete the diffusion process for a training iteration. Next, we feed the noisy textures to the U-Net model which predicts the clean version of the input. Finally, the octree is reversed to the point cloud with the predicted textures to compute loss with the ground truth.

Inference. As illustrated in Fig. 2, we assign pure noise sampled from $\mathcal{N}(0,1)$ to each point of a known point cloud. Then, we amplify the noises using $t = 1000$ and construct an octree based on the point cloud. This tree, along with its noisy textures, is then passed to the U-Net. We denoise the noisy input in a finite number of steps, using the DDPM sampling method developed in [17]. In the end, we reverse the octree to point cloud, then obtain the generated realistic textural surface using the tools provided with [34].

Conditional generalization. To enable conditional generation, we develop an octree-based multi-head cross-attention mechanism based on [36]. The module takes $X \in \mathbb{R}^{N\times C}$ as input and partitions it into $Y \in \mathbb{R}^{\frac{N}{T}\times T\times C}$ patches where N, T, and C mean the number of tree nodes within a batch, the patch size, and the channel number. Then, it computes the query vector $Q \in \mathbb{R}^{\frac{N}{T}\times T\times H\times \frac{H}{C}}$ for H heads utilizing a linear layer. To compute the key vector K and the value vector V from a text / image feature extracted by a CLIP-model [26], we employed a set of two linear layers for each. K and V are then reshaped to match Q's shape for the attentive computation [36].

4. Experiments

4.1. Experimental Setup

Datasets. We conduct experiments on ShapeNet dataset [6] for our experiments, focusing on object categories such as chair, table, car, and bench. The training split follows the approach described in a previous work [43]. To prepare the data for training, we preprocess the dataset to obtain the point cloud and the corresponding ground-truth texture, resulting in point-texture pairs. Specifically, we utilize mesh-sampling [20] to sample 100K points along with their corresponding texture from the provided mesh.

Implementation Details. To train the diffusion models, we organize the point cloud and ground-truth texture into an octree structure with a tree depth of 12. The models are then trained for $2,000$ epochs, employing the AdamW optimizer with a fixed learning rate of $1e-4$ and a batch size of 128. In our training process, we follow the approach suggested by [1, 43] by predicting the clean signals instead of the noise components, which leads to more stable training.

Baselines. We conducted comparisons between our method and several state-of-the-art methods in the context of unconditional generation, namely Texturify [32], Texture Fields [24], Point-UV [43] and Text2Tex [8]. When evaluating Text2Tex [8] in the context of unconditional generation, we employ a general prompt in the form of a "category" for all samples within a specific category. Furthermore, we also compared our method with Point-UV [43] and Text2Tex [8] in the context of text-conditional generation.

Evaluation metrics. We utilize commonly used metrics for evaluating the performance of generative models, specifically, the Fréchet Inception Distance (FID) [16] and Kernel Inception Distance (KID) [3], which measure image quality and diversity.

4.2. Unconditional Generation

We showcase our unconditional texture generation results on ShapeNet [6]. Figure 4 illustrates the remarkable performance of our method. The textures are realistic and maintain the local 3D consistency.

Quantitative evaluations. We quantitatively evaluate our method with comparisons against the state-of-the-art methods. For a fair comparison, all experiments follow the standard evaluation protocol, *i.e.*, FID [16]and KID [3]. Specifically, we render images of the textured shapes produced by each method at a resolution of 512×512 from 4 random viewpoints. It is important to note that the generation time for a single object in Text2Tex [8] is approximately 15 minutes. Due to this time constraint, we randomly selected 150 samples from each category in the test set for evaluation. To ensure a fair comparison, our method utilized the same settings as Text2Tex [8], *i.e.*, randomly rendering 20 views for evaluation. We use a "*" as the

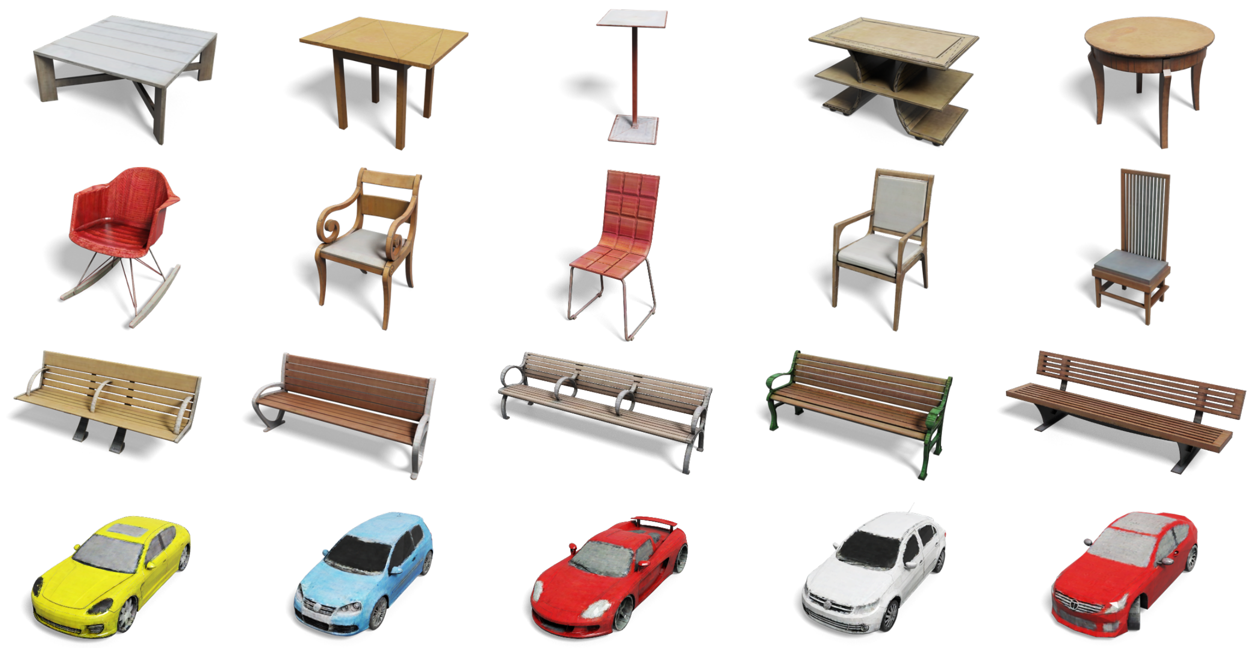
Figure 4. The unconditional texture generation of TexOct. The results demonstrate that TexOct can generate realistic and diverse textures.

Methods	Average		Chair		Car		Table		Bench	
	FID↓	KID↓	FID↓	KID↓	FID↓	KID↓	FID↓	KID↓	FID↓	KID↓
Texture Fields [24]	70.07	5.47	24.24	1.07	156.38	13.64	68.96	4.2	62.71	2.96
Texturify [32]	-	-	28.80	1.32	73.16	4.71	-	-	-	-
Point-UV (1-Stage) [43]	57.69	4.20	17.88	0.77	171.44	15.31	15.94	0.52	25.30	0.21
Point-UV (2-Stage) [43]	17.31	0.30	9.88	0.22	26.89	0.68	9.63	0.15	23.09	0.15
Ours	**14.75**	**0.13**	**9.46**	**0.18**	**21.53**	**0.10**	**7.92**	**0.09**	**20.08**	**0.13**
Text2Tex* [8]	53.22	0.79	56.89	0.86	46.91†	0.44†	55.86	0.98	53.82	0.88
Ours*	**38.11**	**0.14**	**46.14**	**0.27**	**28.13**	**0.07**	**43.86**	**0.16**	**34.31**	**0.07**

Table 1. Comparison against state-of-the-art methods on ShapeNet [6]. We report the FID and KID ($\times 10^2$) metrics. "*" denotes that we randomly select 20 samples from each category and render 20 views for evaluation. "†" denoets the results provided in their paper.

identification. The comparable results are summarized in Table 1, from which we draw the following observation.

Across all categories, our method obtains competitive performance in terms of FID [16] and KID [3]. Notably, it outperforms the leading competitor, Point-UV (stage-2) [43], by an average of 2.56 in FID and 0.17 in KID. Similarly, our method surpasses Text2Tex [8] by an average improvement of 15.11 in FID and 0.65 in KID. These improvements indicate that our method excels at generating highly realistic textures. Furthermore, we compare our method and Point-UV (1-Stage) [43], which utilize 4096 points resulting in rough textures. Our method exhibits significant advantages in both two indicators. This can be attributed to its effective utilization of a dense point cloud to generate finer textures within a single stage.

Method	Average	Chair	Car	Table	Bench
Point-UV [43]	19.3%	12.2%	33.4%	17.8%	13.8%
Ours	80.7%	87.8%	66.6%	82.2%	86.2%

Table 2. Percentage of preference for two methods in a user study. Users more favor our method.

User study. We conducted a user study to assess the quality of the synthesized textures. We randomly present users with 25 pairs of renders for each category, comparing textures generated by Point-UV [43] and our method. In total, there are 100 pairs. Participants are requested to choose the one that is more realistic and finer. We collect a total of 2000 responses from 20 users. The preferences gathered from the study are summarized in Table 2. Comparing our method to Point-UV [43], we observe that our method is preferred by users, with an average preference rate of 80.7%. This result highlights the effectiveness of our approach in generating high-quality textures.

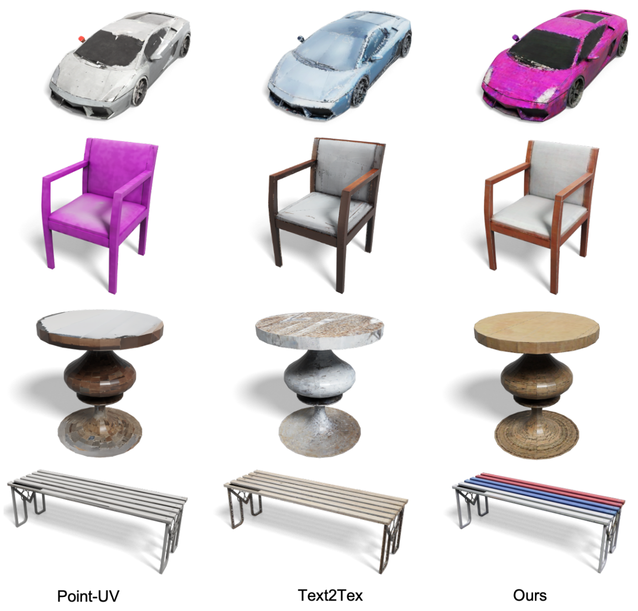

Figure 5. Qualitative comparison with the state-of-the-art method. Our method generates realistic and diverse textures.

Qualitative evaluations. We compare our qualitative results on ShapeNet [6] against the state-of-the-art method in Figure 5. Our method not only generates detailed and realistic textures but also exhibits a rich diversity. For example, in the case of bench, our method generates textures with a rich variety of colors, resulting in a distinct boundary between the bench surface and its legs.

4.3. Text-conditional Generation

We showcase our method's ability to generate conditioned textures using text prompts. We conduct experiments specifically on the chair and table categories. The text-shape pairs provided in [9] serve as our dataset. To incorporate condition-specific information into the network, we leverage the pre-trained vision-language model CLIP [26] to extract the corresponding embedding from the text prompt. This ensures that the specific textual context during texture generation informs our network.

Figure 6 demonstrates the impressive performance of our method in generating textures that closely align with the given text descriptions. The generated textures accurately capture the desired visual attributes specified in the text. This showcases the effectiveness and capability of our approach in synthesizing textures that represent the intended visual characteristics.

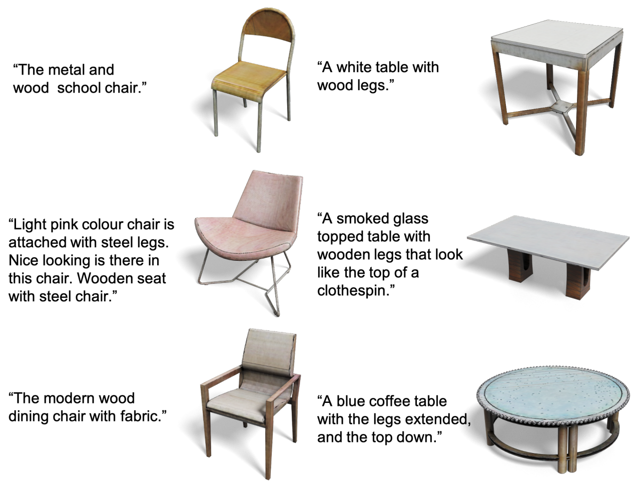
"The metal and wood school chair."
"A white table with wood legs."
"Light pink colour chair is attached with steel legs. Nice looking is there in this chair. Wooden seat with steel chair."
"A smoked glass topped table with wooden legs that look like the top of a clothespin."
"The modern wood dining chair with fabric."
"A blue coffee table with the legs extended, and the top down."

Figure 6. The text-conditional texture generation of TexOct. The generated texture effectively captures the semantic information of the text prompt.

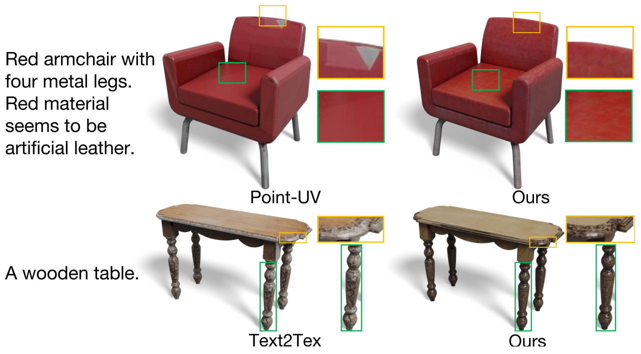
Red armchair with four metal legs. Red material seems to be artificial leather.
Point-UV
Ours
A wooden table.
Text2Tex
Ours

Figure 7. Comparison with the state-of-the-art method for text-conditional generation. Our method generates high-quality and high-frequency textures.

In Figure 7, we compare our qualitative results against text-driven baselines. In comparison with Point-UV [43] and Text2Tex [8], our method generates finer and more consistent textures. The textures generated by the compared methods exhibit local texture inconsistency, blurriness, and a lack of high-frequency patterns. Our method excels in synthesizing more 3D consistent textures with cleaner and more local details.

Depth	Training time	Rec-error	FID↓	KID↓
10	5	0.020	26.06	0.30
11	7	0.011	23.21	0.21
12	19	0.006	23.67	0.13
13	57	0.002	28.19	0.48

Table 3. Evaluation on bench under different octree depth. We evaluate the training time (second/epoch), Reconstruction error (Rec-error), FID and KID.

4.4. Image-conditional Generation

In this section, we additionally showcase our method's ability to generate textures conditioned on a single-view image. We conduct our experiments on the chair and table categories. For the image condition, we randomly render a view from the ground-truth mesh [43]. To infuse the network with image-specific information, we use the pre-trained vision-language model CLIP [26] to extract the corresponding embedding form the given image. Please refer to the supplementary material for details.

4.5. Analysis of Hyper-parameters

Representing point clouds as octrees is a key component of TexOct. As outlined in Section 3.2, the resolution of the octree is determined by the depth parameter L. On the other hand, the representation of 3D objects is influenced by the number of sampled points. In this section, we explore the effects of octree depth and the number of sampled points on the texture generation process. Specifically, we conduct experiments on *bench* category from ShapeNet [6]

Octree depth. In Table 3, we keep the number of sampled points fixed at 100K and vary the octree depth from 10 to 13. We make three observations.

Firstly, we observe that as the depth of the octree increases, the training time for one epoch also increases significantly. This is because the number of layers in the octree grows exponentially, resulting in longer training times.

Secondly, the reconstruction error decreases as the octree depth increases. This indicates that a higher depth allows for a more detailed representation of the point cloud, leading to a more accurate reconstruction.

Lastly, we observe that as the depth increases, the FID and KID initially decrease and then increase. This trend indicates that the octree depth has a two-fold impact on the TexOct. On the one hand, increasing the depth approximately results in a higher resolution, enabling the model to learn more realistic and distinct textures. This can be beneficial in capturing finer details and improving the quality of the generated textures. On the other hand, an overly large depth may cause the model to over-fit the training set, leading to a decrease in generalization performance. This suggests a trade-off between the depth of the octree and the model's ability to generalize to unseen data.

Based on the above observations, we comprehensively consider training time, reconstruction error, and performance of texture generation and ultimately chose to set the depth to 12. This depth setting is consistently applied to other categories.

Point number. In Figure 8, we keep the octree depth fixed at 12 and increase the sampled point number from 10K to 200K. We draw the following observations. When the number of sampled points is set to 10K, we observe that the octree size is the smallest compared to other scenarios.

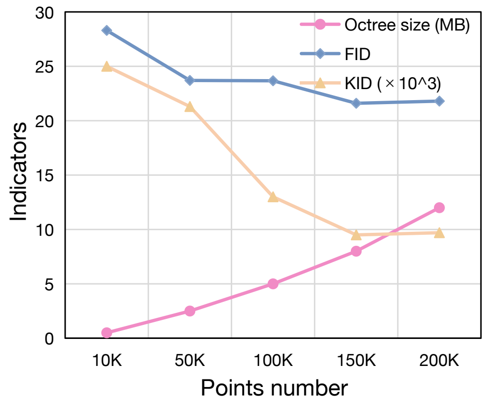

Figure 8. Evaluation on bench. We analyze our method's average octree size (MB), FID, and KID($\times 10^3$) under various numbers of sampled points.

However, the performance in terms of FID and KID metrics is relatively lower compared to other cases. It indicates that 10K points is insufficient to accurately capture the intricate surface details of 3D models, leading to subpar texture generation. As the number of sampled points increases, we observe a gradual decrease in the FID and KID metrics until they converge. However, the octree size also increases rapidly. This indicates that a larger number of points allows for a more precise representation of the 3D object's surface. Nevertheless, continuously increasing the number of points can only lead to limited performance improvements, but it significantly increases the cost of building an octree, specifically by consuming more GPU memory. Finally, we set the number of sampled points to 100K.

5. Conclusion

This paper presents a novel approach for directly generating 3D textures within 3D space. Our method begins by representing the point cloud as an octree structure, which effectively leverages the abundant surface data of 3D objects to capture 3D consistency accurately. Then, we propose an octree-based diffusion model called TexOct, which performs DDPM process on the constructed octree. TexOct gradually reduces the texture noise of the octree node and synthesizes a realistic texture that aligns with the 3D model. The experimental results demonstrate that our method excels in both unconditional and text-conditional texture generation, yielding high-quality 3D textures that are well-received by users.

Limitation. In TexOct, we utilize dense point clouds to generate high-quality texture for 3D models. However, introducing some additional 3D information may further enhance the performance, such as normal, curvature, Laplace–Beltrami operator, *etc.* We will explore these aspects in future work.

References

[1] Titas Anciukevičius, Zexiang Xu, Matthew Fisher, Paul Henderson, Hakan Bilen, Niloy J Mitra, and Paul Guerrero. Renderdiffusion: Image diffusion for 3d reconstruction, inpainting and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12608–12618, 2023. 5

[2] Anand Bhattad, Aysegul Dundar, Guilin Liu, Andrew Tao, and Bryan Catanzaro. View generalization for single image textured 3D models. In *CVPR*, 2021. 2

[3] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018. 5, 6

[4] Alexey Bokhovkin, Shubham Tulsiani, and Angela Dai. Mesh2tex: Generating mesh textures from image queries. *arXiv preprint arXiv:2304.05868*, 2023. 1, 2

[5] Tianshi Cao, Karsten Kreis, Sanja Fidler, Nicholas Sharp, and Kangxue Yin. Texfusion: Synthesizing 3d textures with text-guided image diffusion models. In *ICCV*, pages 4169–4181, 2023. 1, 2

[6] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5, 6, 7, 8

[7] Bindita Chaudhuri, Nikolaos Sarafianos, Linda Shapiro, and Tony Tung. Semi-supervised synthesis of high-resolution editable textures for 3d humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7991–8000, 2021. 2

[8] Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. *arXiv preprint arXiv:2303.11396*, 2023. 1, 2, 5, 6, 7, 3

[9] Kevin Chen, Christopher B Choy, Manolis Savva, Angel X Chang, Thomas Funkhouser, and Silvio Savarese. Text2shape: Generating shapes from natural language by learning joint embeddings. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*, pages 100–116. Springer, 2019. 2, 7

[10] Zhiqin Chen, Kangxue Yin, and Sanja Fidler. Auv-net: Learning aligned uv maps for texture transfer and synthesis. In *CVPR*, pages 1465–1474, 2022. 1, 2

[11] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022. 1

[12] Jun Gao, Wenzheng Chen, Tommy Xiang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3d reconstruction. *Advances In Neural Information Processing Systems*, 33:9936–9947, 2020. 2

[13] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017. 1

[14] Anchit Gupta, Wenhan Xiong, Yixin Nie, Ian Jones, and Barlas Oğuz. 3dgen: Triplane latent diffusion for textured mesh generation. *arXiv preprint arXiv:2303.05371*, 2023. 2

[15] Paul Henderson, Vagia Tsiminaki, and Christoph Lampert. Leveraging 2D data to learn textured 3D mesh generation. In *CVPR*, 2020. 2

[16] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 5, 6

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 3, 5, 1

[18] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 1

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 3, 1

[20] Davi Lazzarotto and Touradj Ebrahimi. Sampling color and geometry point clouds from shapenet dataset. *arXiv preprint arXiv:2201.06935*, 2022. 3, 5

[21] George Kiyohiro Nakayama, Mikaela Angelina Uy, Jiahui Huang, Shi-Min Hu, Ke Li, and Leonidas Guibas. Difffacto: Controllable part-based 3d point cloud generation with cross diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14257–14267, 2023. 3

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

[23] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 3, 1

[24] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4531–4540, 2019. 5, 6

[25] Dario Pavllo, Graham Spinks, Thomas Hofmann, Marie-Francine Moens, and Aurelien Lucchi. Convolutional generation of textured 3D meshes. 2020. 2

[26] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 5, 7, 8, 3

[27] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1 (2):3, 2022. 1

[28] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. In *ACM SIGGRAPH 2023 conference proceedings*, 2023. 2

[29] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022. 1

[30] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022. 1

[31] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 conference proceedings*, pages 1–10, 2022. 1

[32] Yawar Siddiqui, Justus Thies, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Texturify: Generating textures on 3d shape surfaces. In *ECCV*, pages 72–88. Springer, 2022. 1, 2, 5, 6

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

[34] Jiaxiang Tang, Hang Zhou, Xiaokang Chen, Tianshu Hu, Errui Ding, Jingdong Wang, and Gang Zeng. Delicate textured mesh recovery from nerf via adaptive surface refinement. *arXiv preprint arXiv:2303.02091*, 2022. 5

[35] Hao Wang, Guosheng Lin, Steven CH Hoi, and Chunyan Miao. Cycle-consistent inverse gan for text-to-image synthesis. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 630–638, 2021. 1

[36] Peng-Shuai Wang. Octformer: Octree-based transformers for 3D point clouds. *ACM Transactions on Graphics (SIGGRAPH)*, 42(4), 2023. 5

[37] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG)*, 36(4):1–11, 2017. 2, 5

[38] Lemeng Wu, Dilin Wang, Chengyue Gong, Xingchao Liu, Yunyang Xiong, Rakesh Ranjan, Raghuraman Krishnamoorthi, Vikas Chandra, and Qiang Liu. Fast point cloud generation with straight flows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9445–9454, 2023. 3

[39] Rundi Wu, Ruoshi Liu, Carl Vondrick, and Changxi Zheng. Sin3dm: Learning a diffusion model from a single 3d textured shape. *arXiv preprint arXiv:2305.15399*, 2023. 3

[40] Zijie Wu, Yaonan Wang, Mingtao Feng, He Xie, and Ajmal Mian. Sketch and text guided diffusion model for colored point cloud generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8929–8939, 2023. 3

[41] Fanbo Xiang, Zexiang Xu, Milos Hasan, Yannick Hold-Geoffroy, Kalyan Sunkavalli, and Hao Su. Neutex: Neural texture mapping for volumetric neural rendering. In *CVPR*, pages 7119–7128, 2021. 2

[42] Kangxue Yin, Jun Gao, Maria Shugrina, Sameh Khamis, and Sanja Fidler. 3dstylenet: Creating 3d shapes with geometric and texture style variations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12456–12465, 2021. 2

[43] Xin Yu, Peng Dai, Wenbo Li, Lan Ma, Zhengzhe Liu, and Xiaojuan Qi. Texture generation on 3d meshes with point-uv diffusion. In *ICCV*, pages 4206–4216, 2023. 1, 2, 3, 5, 6, 7, 8

[44] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 3

[45] Xin-Yang Zheng, Hao Pan, Peng-Shuai Wang, Xin Tong, Yang Liu, and Heung-Yeung Shum. Locally attentional sdf diffusion for controllable 3d shape generation. *ACM Transactions on Graphics (SIGGRAPH)*, 42(4), 2023. 3

[46] Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Non-stationary texture synthesis by adversarial expansion, 2018. 1