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

Supplementary Material

1. Algorithm Details

Forward process. The Denoising Diffusion Probabilistic Model (DDPM) [23] is a framework that learns the data distribution by incorporating a sequence of latent variables and matching the joint distribution.

In our method, we apply the DDPM process to the octree nodes. The model begins with a sample from the data distribution, denoted as $x_0 \sim q(\mathbf{x}_0)$. Through a forward process, $q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)=\prod_{t=1}^{T} q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)$, the data is progressively perturbed using Gaussian kernels $q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right) := \mathcal{N}\left(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right)$. This process generates a sequence of increasingly noisy latent variables $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T$. Importantly, x_t can be directly sampled from x_0. Additionally, the shape of each x_t aligns with the octree structure.

$$q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)=\mathcal{N}\left(\mathbf{x}_t ; \sqrt{\bar{\alpha}_t} \mathbf{x}_0,\left(1-\bar{\alpha}_t\right) \mathbf{I}\right), \tag{6}$$

where $\alpha_t := 1-\beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$. In the forward process, the variances β_t are generally fixed and increase linearly from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$. It is important to choose a sufficiently large value for T (e.g., 1000) to ensure that $q\left(\mathbf{x}_T \mid \mathbf{x}_0\right)$ approximates a standard normal distribution $\mathcal{N}(0, \mathbf{I})$.

The primary objective of the diffusion model is to model the joint distribution $q\left(\mathbf{x}_{0:T}\right)$, which encompasses a tractable sampling path for the marginal distribution $q\left(\mathbf{x}_0\right)$.

Reverse process. To learn how to reverse the forward process, the diffusion model defines a parameterized Markov chain with parameterized transition kernels:

$$\begin{aligned} p_\theta\left(\mathbf{x}_{0:T}\right) &:= p\left(\mathbf{x}_T\right) \prod_{t=1}^{T} p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right), \\ p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) &:= \mathcal{N}\left(\mathbf{x}_{t-1} ; \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t\right)\right), \end{aligned} \tag{7}$$

where $\boldsymbol{\mu}_\theta$ and $\boldsymbol{\Sigma}_\theta$ are optimized. The denoising model is trained with the variational lower bound of the log-likelihood.

$$\mathrm{E}_{q\left(\mathbf{x}_0\right)}\left[-\log p_\theta\left(\mathbf{x}_0\right)\right] \leq \mathrm{E}_{q\left(\mathbf{x}_{0: T}\right)}\left[-\log \frac{p_\theta\left(\mathbf{x}_{0: T}\right)}{q\left(\mathbf{x}_{1: T} \mid \mathbf{x}_0\right)}\right] =: L. \tag{8}$$

The loss term L can be rewritten as Eq 1 in manuscript.

$$\mathcal{L} = -p_\theta(x_0|x_1) + \sum_t \mathcal{D}_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)) \tag{9}$$

Both two terms compared in the KL divergence are Gaussians, *i.e.*,:

$$\begin{aligned} q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) &= \mathcal{N}\left(\mathbf{x}_{t-1} ; \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right), \tilde{\beta}_t \mathbf{I}\right), \\ p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) &:= \mathcal{N}\left(\mathbf{x}_{t-1} ; \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t\right)\right), \end{aligned} \tag{10}$$

where $\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$ and $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$. DDPM fix $\boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t\right) = \sigma_t^2 \mathbf{I}$ during training, where σ_t^2 is set to be β_t or $\tilde{\beta}_t$. In our experiments, we set $\sigma_t^2 = \beta_t$.

Optimization. Models can be trained in the reverse process to predict the mean value of $\mathbf{x}_{t-1}$, *i.e.* $\boldsymbol{\mu}_t$. Alternatively, by modifying the parameterization, it is also possible to train the model to predict $\mathbf{x}_0$ or $\boldsymbol{\epsilon_t}$, as demonstrated in [17]. The original DDPM [17] predicts $\boldsymbol{\epsilon_t}$. In this case, the training term is performed as follows, utilizing the reparameterization trick [19] and empirical simplification [17]:

$$L_{\text{simple}} := \mathbb{E}_{t, \mathbf{x}_0, \epsilon}\left[\left\|\boldsymbol{\epsilon_t} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|_2^2\right], \tag{11}$$

where $\epsilon_t \sim \mathcal{N}(0, \mathbf{I})$ and t is uniformly sampled between 1 and T.

In our experiments, we predict $\mathbf{x}_0$ for more stable training. The loss function is modified as:

$$L_{\mathrm{x}_0} := \mathbb{E}_{t, \mathbf{x}_0, \epsilon}\left[\left\|\mathbf{x}_0 - Gen_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|_2^2\right], \tag{12}$$

where Gen_θ is the network to be optimized.

Inference. After training, started from an initial noise map $x_{t_{\text{max}}} \sim \mathcal{N}(0, \mathbf{I})$, new textures can be then generated via iteratively sampling from $p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)$, using the following equation:

$$\mathbf{x}_{t-1} = \tilde{\boldsymbol{\mu}}_t + \sigma_t \mathbf{z}, \tag{13}$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. In particular, we predict the x_0, so that the sampling process can be modified as:

$$\begin{aligned} \mathbf{x}_{t-1} =& \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\hat{\mathbf{x}}_0 + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}\mathbf{x}_t + \sigma_t \mathbf{z} \\ =& \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} G_\theta\left(\mathbf{x}_t, t\right) + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}\mathbf{x}_t + \sigma_t \mathbf{z}. \end{aligned} \tag{14}$$

Figure 9. The image-conditional texture generation of our method. Our method is adaptable to craft textures guided by singe-view images. The gray boxes represent image conditions.

2. Network Architecture

We construct TexOct as a U-Net structure, where the initial input tensor is denoted as $\mathbf{x}_0 \in \mathbb{R}^{N\times 3}$, with N representing the number of octree nodes. We first employ an input-embedding layer (Octree convolution) that transforms it into a tensor $\mathbf{x}_{\text{input}} \in \mathbb{R}^{N\times 32}$. Then, the input tensor $\mathbf{x}_{\text{input}}$ is passed through the U-Net architecture. The channel numbers for each level of the U-Net are specified in Table 4. Finally, the output of U-Net is passed through an out-embedding layer to obtain the final output $\mathbf{x}_{\text{out}} \in \mathbb{R}^{N\times 3}$.

Block Type	Level	In-Channels	Out-Channels
Encoder	0	32	32
	1	32	64
	2	64	128
	3	128	128
Middle	0	128	128
	1	128	128
Decoder	0	128	128
	1	128	64
	2	64	32
	3	32	32

Table 4. Details of TexOct.

3. User Study Details

We randomly select 25 pairs of renders for each category, comparing Point-UV [43] and our method. In total, there are 100 pairs. To better visualize the samples, we render multi-view images for those objects from 4 preset viewpoints. After the samples are prepared, we ask the participants to pick the sample from those pairs that is more realistic and finer. To avoid biases and cheating in this user study, we shuffle the pairs so that there is no positional hint of our method. In the end, we gather 2000 responses from 20 participants to calculate the preferences.

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-

Methods	Average		Text-Chair		Text-Table		Image-Chair		Image-Table	
	FID↓	KID↓	FID↓	KID↓	FID↓	KID↓	FID↓	KID↓	FID↓	KID↓
Point-UV (1-Stage)* [43]	22.55	0.98	18.05	0.71	17.77	0.60	23.64	1.18	30.72	1.44
Point-UV (2-Stage)* [43]	10.12	0.21	8.28	0.15	9.16	0.10	11.49	0.37	11.56	0.21
Ours	**7.54**	**0.08**	**8.11**	**0.11**	**7.54**	**0.06**	**7.85**	**0.11**	**6.67**	**0.04**

Table 5. Comparison against Point-UV [43] in the context of conditional generation. "*" denotes that we obtain the results by using the official code.

Figure 10. Additional results of text-conditional generation.

trained vision-language model CLIP [26] to extract the corresponding embedding form the given image. Figure 9 demonstrates that our method succeeds in generating textures that align well with the given images.

5. Additional Results

In order to further showcase the effectiveness of our method, we present additional results.

Quantitative evaluations. Additionally, in the context of conditional generation, we conduct a quantitative comparison between our method and Point-UV [43]. Specifically, we adopt a unified evaluation protocol as described in Section 4.2. We report the FID and KID in Table 5. Our method outperforms Point-UV [43] by a clear margin. These improvements indicate that our method excels at generating highly realistic textures.

Qualitative evaluations. We provide additional results of text-conditional and unconditional generation in Figure 10 and Figure 11, respectively. The generated results further validate the effectiveness of our method. The texture produced by our approach is detailed and realistic, aligning closely with the text descriptions in context of text-conditional generation.

Inference time. We conduct a comparison of the inference time between Point-UV [43], Text2Tex [8], and our method. The evaluation is performed under identical hardware conditions, specifically using a Tesla A-100 GPU. We focus solely on the time required to generate a single instance of the final texture, excluding any time spent on saving intermediate results. Specifically, Point-UV [43] exhibits an inference time of 30 seconds, while Text2Tex [8] requires 6 minutes. In contrast, our method achieves the same task in just 5 seconds, showcasing its efficiency.

Figure 11. Additional results of unconditional generation.