International Journal of Computer Vision (2025) 133:3105–3128
https://doi.org/10.1007/s11263-024-02326-x

SLIDE: A Unified Mesh and Texture Generation Framework with Enhanced Geometric Control and Multi-view Consistency

Jinyi Wang[1] · Zhaoyang Lyu[2] · Ben Fei[3] · Jiangchao Yao[1] · Ya Zhang[2] · Bo Dai[2] · Dahua Lin[3] · Ying He[4] · Yanfeng Wang[2]

Received: 1 April 2024 / Accepted: 1 December 2024 / Published online: 23 December 2024
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2024

Abstract
The generation of textured mesh is crucial for computer graphics and virtual content creation. However, current generative models often struggle with challenges such as irregular mesh structures and inconsistencies in multi-view textures. In this study, we present a unified framework for both geometry generation and texture generation, utilizing a novel sparse latent point diffusion model that specifically addresses the geometric aspects of models. Our approach employs point clouds as an efficient intermediate representation, encoding them into sparse latent points with semantically meaningful features for precise geometric control. While the sparse latent points facilitate a high-level control over the geometry, shaping the overall structure and fine details of the meshes, this control does not extend to textures. To address this, we propose a separate texture generation process that integrates multi-view priors post-geometry generation, effectively resolving the issue of multi-view texture inconsistency. This process ensures the production of coherent and high-quality textures that complement the precisely generated meshes, thereby creating visually appealing and detailed models. Our framework distinctively separates the control mechanisms for geometry and texture, leading to significant improvements in the generation of complex, textured 3D content. Evaluations on the ShapeNet dataset for geometry and the Objaverse dataset for textures demonstrate that our model surpasses existing methods in terms of geometric quality, control, and the generation of coherent, high-quality textures.

Keywords Mesh generation · Latent diffusion model · Texture generation · Controllable generation

1 Introduction

The increasing demand for diverse, high-quality 3D content across various industries, such as gaming, architecture, and social media, underscores the inefficiency of manual asset creation. This process is not only time-consuming and technically demanding but also labor-intensive and costly. In

Communicated by Shengfeng He.

Jinyi Wang, Zhaoyang Lyu and Ben Fei have contributed equally to this work.

Jinyi Wang, Zhaoyang Lyu and Ben Fei are co-first authors.

✉ Jiangchao Yao
sunarker@sjtu.edu.cn

✉ Yanfeng Wang
wangyanfeng@sjtu.edu.cn

Jinyi Wang
jinyi.wang@sjtu.edu.cn

Zhaoyang Lyu
lvzhaoyang@pjlab.org.cn

Ben Fei
benfei@cuhk.edu.hk

Ya Zhang
ya_zhang@sjtu.edu.cn

Bo Dai
daibo@pjlab.org.cn

Dahua Lin
dhlin@ie.cuhk.edu.hk

Ying He
yhe@ntu.edu.sg

[1] Cooperative Medianet Innovation Center, Shanghai Jiao Tong University, Shanghai, China

[2] School of Artificial Intelligence, Shanghai Jiao Tong University, Shanghai, China

[3] Department of Information Engineering, The Chinese University of Hong Kong, Sha Tin, Hong Kong, China

[4] College of Computing and Data Science, Nanyang Technological University, Singapore, Singapore

gaming, the need for numerous assets like characters and furniture is ever-growing, and the aesthetic quality of these assets can significantly impact the immersive nature of the gaming environment. In architecture, precise and detailed models of buildings and structures are essential for visualization, simulation, and planning. Meanwhile, social media platforms are increasingly leveraging 3D content for augmented reality (AR) and virtual reality (VR) experiences, allowing users to interact with enhanced digital environments. However, the realism of these 3D models often hinges on detailed mesh representations, including vertices, edges, faces, and textures. This reality amplifies the urgency of automating the production of controllable, high-quality textured meshes.

Some methods, including GET3D (Gao et al., 2022) and 3DGen (Gupta et al., 2023), simultaneously generate geometries and textures. These methods employ triplanes as an intermediate representation for generating textured meshes, with GET3D utilizing a Generative Adversarial Network (GAN) and 3DGen adopting a diffusion model. However, the concurrent generation of geometries and textures may lead to models that disproportionately favor texture detailing over geometric accuracy, culminating in 3D shapes with compromised geometries obscured by textures. Therefore, a disentangled approach to the generation of geometries and textures can be more robust and efficient. This perspective is underpinned by two considerations: Firstly, in the film and gaming industries, it is customary for users and designers to apply textures to 3D assets post the generation of their geometry, thereby ensuring that the texturing complements the geometrically optimized models. Secondly, methods like GET3D and 3DGen, which generate structure and texture concurrently, are prone to producing textures characterized by coarseness, unrealism, and a lack of diversity, a consequence of training on the ShapeNet dataset with its inherent textural limitations. In light of these observations, we propose a two stage method for generating geometry and texture, wherein the initial stage focuses on geometry generation, followed by a subsequent stage dedicated to the synthesis of textures.

Considering these aspects, various methods have been developed to generate 3D shapes devoid of textures, offering potential for enhancing the quality of their geometry. For example, methods such as those reported in Li et al. (2022a), Chou et al. (2022), Shue et al. (2022), Liu et al. (2023b), Zheng et al. (2023), Gupta et al. (2023), Nam et al. (2022) employ implicit fields (Mescheder et al., 2018; Park et al., 2019) for mesh generation. These approaches often involve the design of latent representations, including points, voxels, or triplanes, to depict the implicit fields, subsequently training diffusion models on these latent structures. The reconstruction of meshes is achievable through techniques such as marching cubes (Lorensen & Cline, 1987) or Deep Marching Tetrahedra (DMTet (Shen et al., 2021)). However, despite the advancements, these methods are time-intensive and face challenges in achieving controllable mesh generation. While certain geometries have been produced, their controllability is yet to meet the desired standards, prompting an ongoing effort to improve this aspect.

Moreover, methodologies exist that facilitate texture generation contingent upon geometric configurations. Predominantly, two principal methodologies are utilized for leveraging pre-trained diffusion models to accomplish high-fidelity 3D texture synthesis. The inaugural methodology, termed Score Distillation Sampling (SDS) (Poole et al., 2022), endeavors to refine 3D representations through their alignment with image priors. Nonetheless, this methodology is encumbered by limitations, including augmented color saturation and diminished generative diversity, attributed to its substantial dependency on classifier-free guidance weights. Conversely, an alternate methodology is the generation of 2D imagery from multiple perspectives (Chen et al., 2023a; Richardson et al., 2023), which utilizes iterative projection of views onto the texture map. This is achieved through the application of inpainting and depth-conditioned latent diffusion models, thereby ameliorating the coherence of texture synthesis. However, this approach, centering on the denoising of discrete 2D view images, may engender texture inconsistencies due to alignment discrepancies.

To address these, we propose a two-stage method that separately generate geometry and texture (Fig. 1). In the first stage, our method focuses on geometry generation. We adopt Denoising Diffusion Probabilistic Models (DDPMs) to accurately model the distribution of meshes. To counteract the high computational demands and intricate control issues associated with dense point clouds, we introduce the Sparse Latent poInt Diffusion modEl (SLIDE). This innovative approach consists of a point cloud encoder to convert dense point clouds into a manageable set of sparse latent points, significantly reducing complexity and enhancing control over the mesh structure (Fig. 2). This stage is crucial for establishing a detailed and accurate geometric foundation for subsequent texturing. The second stage of our methodology is devoted to texture application, employing a coarse-to-fine approach. Initially, multi-view diffusion models are integrated to establish texture consistency across diverse viewpoints, enhanced by the incorporation of a depth-conditioned diffusion model for depth-awareness. To address the limitations of resolution and untextured areas, the coarse texture map undergoes refinement. This involves segmenting the map into a refine region and a generation region, where inpainting, denoising, and projection are meticulously applied. Such a way significantly improves the realism and resolution of the textures, culminating in textured meshes that are both high in resolution and consistent across views.

Fig. 1 Textured meshes generated by our method

The proposed method builds upon our early work (Lyuv et al., 2023), which features a parallel task of texture generation. We make several new contributions:

- We propose a novel coarse-to-fine generative framework that is capable of producing high-resolution, view-consistent texture maps for untextured 3D meshes.
- Our texture generation approach begins with a coarse phase that integrates multi-view diffusion models to guarantee texture uniformity across different viewing angles, followed by a refinement phase for inpainting untextured areas and improving texture quality.
- Extensive evaluations on the Objaverse and ShapeNet datasets illustrate the superiority of our method in generating high-fidelity textures that are consistent across all views, marking a significant advancement over existing texture synthesis techniques.

These contributions highlight our method's ability to produce detailed and realistic textures for 3D models, enhancing both the visual quality and the practical utility of generated objects in virtual environments.

2 Related Work

2.1 Mesh Generation

Diffusion Model for 3D Shape Generation Exploration of diffusion models for 3D shape generation has yielded promising results. Initially, these models were applied to the generation of point clouds, with notable contributions from various researchers (Luo & Hu, 2021a; Zhou et al., 2021a; Zeng et al., 2022a; Nichol et al., 2022a). Subsequently, efforts have been made to reconstruct meshes from these point clouds, employing advanced surface reconstruction techniques (Peng et al., 2021b; Zeng et al., 2022a). Parallel to this, an alternative strand of research has focused on mesh generation via implicit fields, as evidenced by a series of studies (Chou et al., 2022; Gupta et al., 2023; Li et al., 2022a; Liu et al., 2023b; Nam et al., 2022; Shue et al., 2022; Zheng et al., 2023). These studies often involve the creation of latent representations-such as points, voxels, or triplanes-to encapsulate the implicit fields, with diffusion models subsequently trained on these representations. The final stage of mesh reconstruction is typically achieved through methods like marching cubes (Lorensen & Cline, 1987) or deep marching tetrahedra (DMTet (Shen et al., 2021)).

2.2 Texture Generation

Traditional texture synthesis methods for 3D assets primarily utilized exemplar patterns and global optimization techniques (Huang et al., 2020; Kopf et al., 2007; Lefebvre & Hoppe, 2006; Turk, 2001; Wei & Levoy, 2001; Wei et al., 2009; Zhou & Koltun, 2014). Recent advancements have shifted towards learning-based approaches, demonstrating enhanced capability in generating textures for complex 3D shapes (Chen et al., 2023d; Karnewar et al., 2023; Li et al., 2023; Pan et al., 2023; Qian et al., 2023; Raj et al., 2023; Tang et al., 2023; Yang et al., 2023; Zhuang et al., 2023).
Iteratively Texturing via 2D Diffusion Models Advances in 2D text-to-image diffusion models (Ramesh et al., 2022; Rombach et al., 2022; Saharia et al., 2022) have paved the way for their adaptation in 3D texturing, with methods like TEXTure and TexFusion iteratively refining textures from multiple viewpoints, improving global coherence (Cao et al., 2023; Richardson et al., 2023). However, challenges in view consistency persist, which our framework addresses by incorporating multi-view priors from multi-view diffusion models to mitigate this issue.
Optimization-based 3D Generation via 2D diffusion model Earlier, optimization-driven methods leveraged CLIP for texture mapping (Hong et al., 2022; Lei et al., 2022; Ma et al., 2023; Michel et al., 2022; Mohammad Khalid et al., 2022), later enhanced by Score Distillation Sampling (SDS) for text-

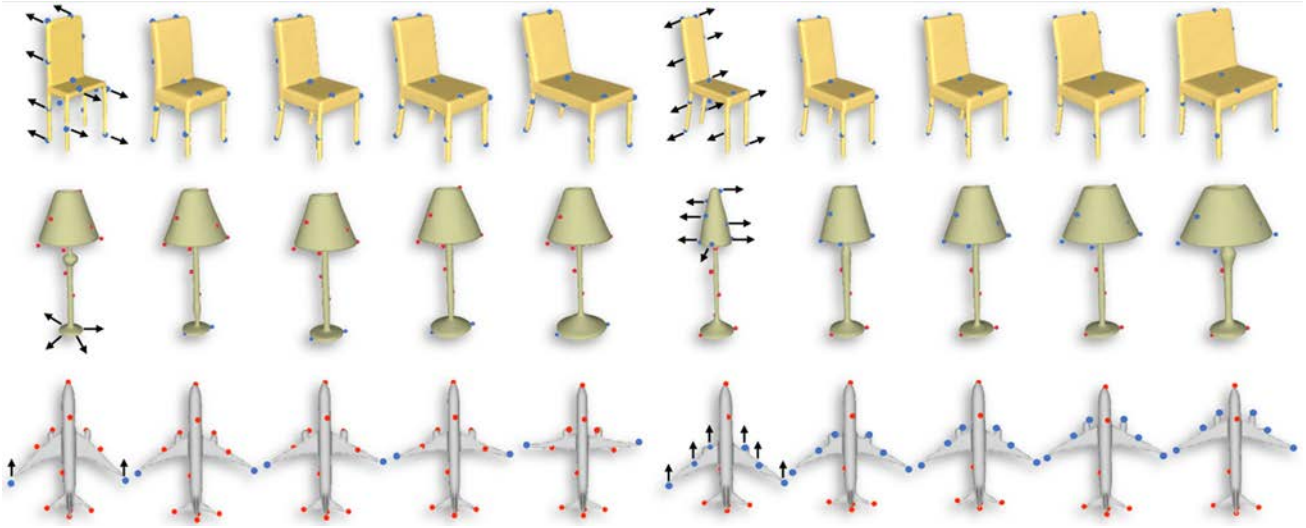


Fig. 2 We can use the sparse latent points to control the shape of the generated meshes. Red points are stationary, and blue points are moving. Black arrows indicate the moving direction of the blue points. Some points are invisible because they are within the mesh. Note that the latent points are not always strictly lying on the surface of the generated meshes. This is because our point cloud decoder assumes that some noises exist in the positions of the sparse latent points. It will generate a mesh that best fits the latent points, but avoid generating defective meshes just to strictly fit the latent points (Color figure online)

to-3D synthesis, yielding multi-view consistent textures but with potential geometric inaccuracies (Chen et al., 2023c; Lin et al., 2023; Metzer et al., 2023; Poole et al., 2022).

Generative Texturing from 3D Data Learning-based methods have evolved to train generative models directly from 3D data, with early techniques focusing on implicit texture fields and recent ones employing mesh-based convolution operators for enhanced texture prediction (Bokhovkin et al., 2023; Chen et al., 2023b; Collins et al., 2022; Deitke et al., 2023; Gao et al., 2022; Gupta et al., 2023; Jun & Nichol, 2023; Li et al., 2022b; Luo et al., 2023; Nichol et al., 2022b; Oechsle et al., 2019; Siddiqui et al., 2022). Despite improvements, challenges in handling object variability across categories persist, highlighting the need for methods like ours that offer a more generalized solution.

2.3 Textured Mesh Generation

Several works have been conducted in the domain of textured mesh generation. Variational Autoencoders (VAEs) (Henderson et al., 2020) and Generative Adversarial Networks (GANs) (Chan et al., 2022) based models have also been used to learn distributions from 3D data for generation. Similar to our approach, these methods focus on directly generating 3D data; however, they utilize VAEs and GANs, whereas our work employs a diffusion model (DDPM). VAE-based methods tend to generate less satisfactory 3D shapes compared with GANs- or DDPM-based methods. The training of GAN-based methods is not stable. Compared with VAE- and GAN-based methods, our DDPM-based method is able to generate high-quality 3D shapes with better diversity.

Several methods (Anciukevičius et al., 2023; Rakotosaona et al., 2024) have leveraged image diffusion techniques that denoise images through 3D rendering to generate 3D objects. These approaches have the advantage of reducing reliance on 3D datasets, which are frequently limited and expensive, by making use of plentiful and well-annotated 2D data. These methods demonstrate excellence in diversity and text-driven control, yet they are hindered by slower generation speeds and the possibility of inconsistencies across multiple views because they rely on optimization-based processes.

Recent advancements in this field have focused on advancing image diffusion techniques while also tackling the challenge of multi-view inconsistency through the simultaneous generation of multi-view consistent images. These methods (Anciukevicius et al., 2024; Höllein et al., 2024; Szymanowicz et al., 2023) significantly improve not only the quality and consistency of the generated views but also the speed of generation, presenting a substantial step forward in the application of 3D generative technologies.

3 Diffusion Model Preliminaries

Denoising Diffusion Probabilistic Models (DDPMs) are generative models that learn the distribution of samples in a dataset. A DDPM is composed of two processes: the diffusion process and the reverse process. The diffusion process gradually adds noise to clean samples $\boldsymbol{x}^0$ and turns them into

Gaussian noises $\boldsymbol{x}^T$ after T steps. It is defined as

$$q(\boldsymbol{x}^1, \cdots, \boldsymbol{x}^T|\boldsymbol{x}^0) = \prod_{t=1}^{T} q(\boldsymbol{x}^t|\boldsymbol{x}^{t-1}), \tag{1}$$

$$\text{where } q(\boldsymbol{x}^t|\boldsymbol{x}^{t-1}) = \mathcal{N}(\boldsymbol{x}^t; \sqrt{1-\beta_t}\boldsymbol{x}^{t-1}, \beta_t \boldsymbol{I}), \tag{2}$$

$\mathcal{N}$ is the Gaussian distribution. In our experiments, we set $T = 1000$, and β_t linearly increase from 1×10^{-4} to 2×10^{-2} as t increases from 1 to T. The reverse process is the data generation process. It starts from a Gaussian noise $\boldsymbol{x}^T$ and denoises it step by step, eventually turning it into a clean sample $\boldsymbol{x}^0$. The reverse process is formally defined as

$$\begin{aligned} p_{\boldsymbol{\theta}}(\boldsymbol{x}^0, \cdots, \boldsymbol{x}^{T-1}|\boldsymbol{x}^T) &= \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}^{t-1}|\boldsymbol{x}^t), \\ \text{where } p_{\boldsymbol{\theta}}(\boldsymbol{x}^{t-1}|\boldsymbol{x}^t) &= \mathcal{N}(\boldsymbol{x}^{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}^t, t), \tilde{\beta}_t \boldsymbol{I}), \end{aligned} \tag{3}$$

$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$. We follow Ho et al. (2020a) to reparameterize the mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}^t, t)$ as

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}^t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}^t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}^t, t)\right), \tag{4}$$

where $\alpha_t = 1-\beta_t$, $\bar{\alpha}_t = \prod_{i=1}^{t}\alpha_i$, and $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ is a neural network with parameters specified by $\boldsymbol{\theta}$. The loss to train the network is

$$L(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}^0\sim p_{\text{data}}} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\sqrt{\bar{\alpha}_t}\boldsymbol{x}^0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2, \tag{5}$$

where p_{data} is the distribution of the dataset, t is sampled uniformly between 1 and T, and $\boldsymbol{\epsilon}$ is a Gaussian noise.

4 Method

Our objective is to produce a textured mesh by separating the process into distinct phases for geometry and texture generation. As illustrated in Fig. 3, we initiate the generation of mesh geometry using $\mathcal{M} = \mathcal{G}(GaussianNoise)$. Subsequently, our method $\mathcal{T}$ is applied to texture the mesh, incorporating input from the prompt $\mathbf{y}$, resulting in the final textured mesh denoted as $\mathcal{M}_{tex} = \mathcal{T}(\mathcal{M}, \mathbf{y})$.

The specifics of the methodologies employed will be elaborated in the subsequent two subsections.

4.1 Geometry Generation

In this subsection, we address the geometric aspect of mesh generation, which poses significant challenges due to the irregular data structure of meshes, consisting of vertices and faces that define their shape and connectivity. While modeling vertex positions is feasible, capturing the complex connections among vertices is more arduous. To circumvent this, we adopt point clouds with normals as an intermediate, simplified representation of meshes, allowing for efficient data handling and processing. In our methodology, point clouds, sampled to represent mesh surfaces with 2048 points, are utilized to model the distribution via established generative models (Cai et al., 2020; Luo & Hu, 2021b; Zhou et al., 2021b).

For mesh reconstruction from these point clouds, the Shape as Points (SAP) approach (Peng et al., 2021a), integrating an upsampling network and Differentiable Poisson Surface Reconstruction (DPSR), is employed. Further details on SAP are available in the original publication or Appendix A.1.

Despite the efficiency of point clouds with normals as a mesh proxy, they often represent 3D shapes redundantly and pose challenges in manipulation and control. Addressing this, we propose encoding point clouds into a set of sparse latent points, each with associated features, to succinctly capture the essence of the shape. This sparser representation, illustrated in Fig. 4a, conceptualizes a 3D shape as a skeleton with features that detail its geometry.

To ensure coverage of the original point cloud, Farthest Point Sampling (FPS) is utilized to select a specified number of latent points (16 in our experiments), initiating with either the centroid or a randomly selected point. Subsequently, a point cloud encoder is developed to attach geometric features to these latent points, while a corresponding decoder reconstructs the original point cloud with normals. The specifics of these encoding and decoding mechanisms are elaborated in the subsequent section.

4.1.1 Point Cloud Autoencoder

As mentioned above, we need a point cloud encoder to encode a point cloud to a sparse set of points with features, and a decoder to decode the sparse latent points back to the input point cloud. In this section, we explain the detailed architectures of the point cloud encoder and decoder.

Point Cloud Encoder The encoder translates a point cloud into features for a Farthest Point Sampling (FPS) selected sparse point set, as depicted in Fig. 5a. It incorporates enhanced Set Abstraction (SA) modules, equipped with an attention mechanism, as introduced in PDR (Lyuv et al., 2021). The SA module, which processes points with associated features, employs FPS for subsampling and feature propagation. It computes the features of subsampled points by locating the K nearest neighbors, applying a shared Multilayer Perceptron (MLP) for feature transformation, and then integrating these features using the attention mechanism, detailed in (Lyuv et al., 2021).

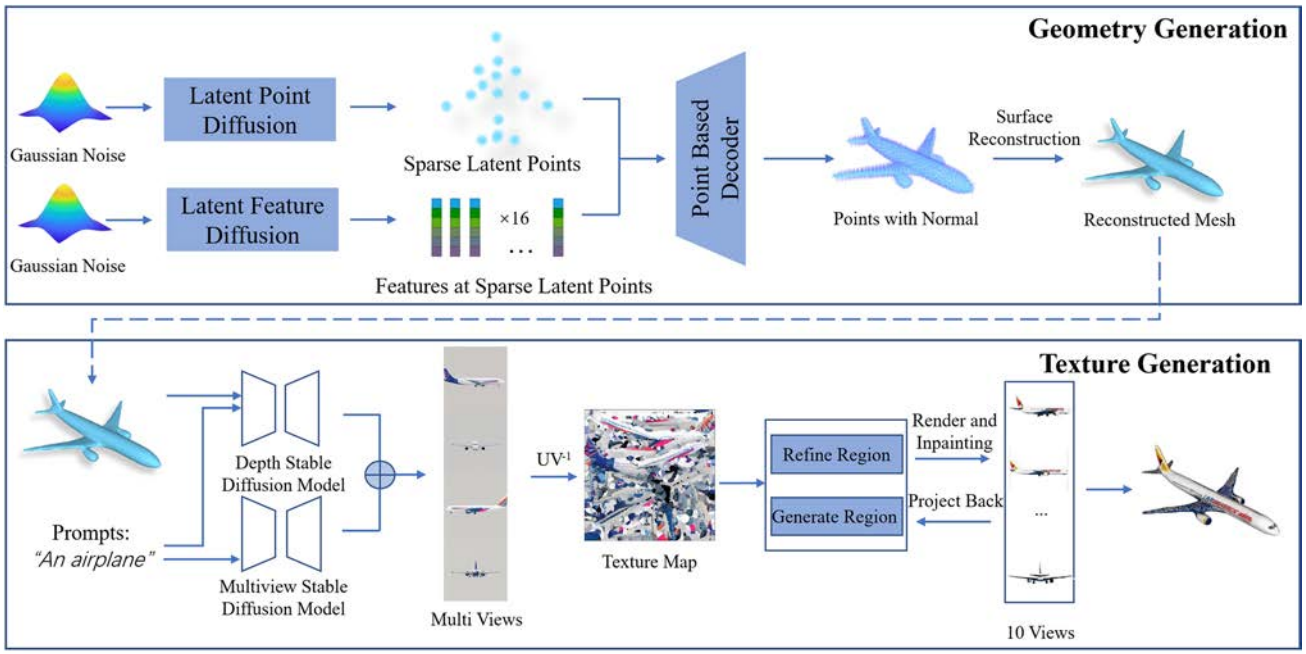


Fig. 3 The pipeline of our method including geometry generation method and texture generation

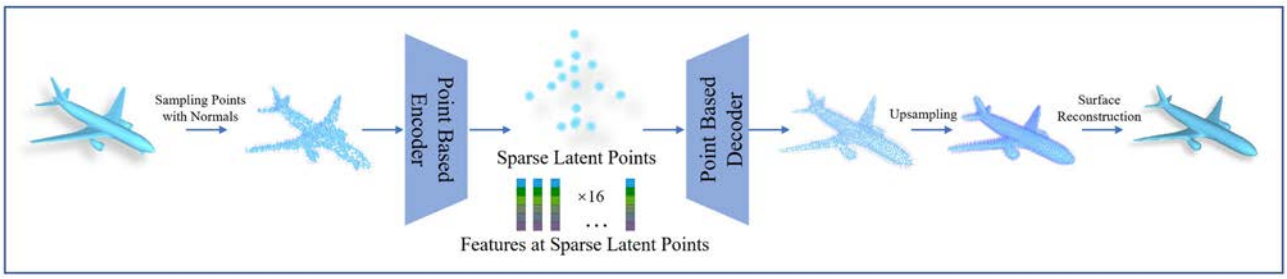
The encoding process involves four sequential SA modules that reduce the point cloud from 2048 points to hierarchical subsets of 1024, 256, 64, and 32 points, thereby refining the features. The final encoding to the sparse latent space of 16 points is facilitated by the Feature Transfer (FT) module from PDR (Lyuv et al., 2021), which parameterizes feature mapping between point sets through a neural network. A lightweight PointNet++ (Qi et al., 2017) is utilized to generate initial features for these sparse latent points, which are then enhanced by the FT module, combining features from the final SA module to yield the point cloud's representative features.

Overall, our encoder, structured as in Fig. 5a, employs a hierarchical feature extraction approach through four SA modules, followed by feature refinement via PointNet++ (Qi et al., 2017) and the FT module for the sparse latent representation.

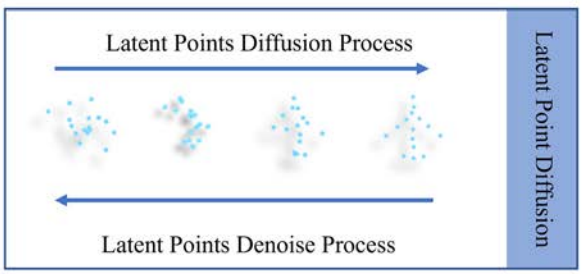
Point Cloud Decoder The point cloud decoder, designed for reconstructing the input point cloud from encoded sparse latent features, comprises a structured assembly visualized in Fig. 5b. Integral to this architecture are three point upsampling (PU) modules, which sequentially expand the number of points from sixteen to two thousand and forty-eight. At each stage l, a PU module processes a point set $\boldsymbol{X}^l = \{x_j^l \in \mathbb{R}^3 | 1 \leq j \leq N^l\}$, with each point x_j^l associated with a feature vector $\boldsymbol{F}^l = \{f_j^l \in \mathbb{R}^{d^l} | 1 \leq j \leq N^l\}$, where N^l denotes the number of points, f_j^l the feature at point x_j^l, and d^l the feature dimension. The first PU module receives as input the sparse latent points $\boldsymbol{X}^1$ along with their features $\boldsymbol{F}^1$. A shared Multi-layer Perceptron (MLP) transforms the feature f_j^l at each point x_j^l into γ displacements, subsequently added to the original point to generate γ new points, thereby achieving an upsampling factor of γ. To ensure uniformity in the distribution of the upsampled points, Farthest Point Sampling (FPS) is employed post-upsampling to reduce the number of points by half, resulting in an effective upsampling factor of $\gamma/2$. This yields the upsampled point set $\boldsymbol{X}^{l+1} = \{x_j^{l+1} \in \mathbb{R}^3 | 1 \leq j \leq N_{l+1}\}$, with $N_{l+1} = \gamma N_l / 2$.

The subsequent PU module ($l+1$-th) engages in further upsampling of $\boldsymbol{X}^{l+1}$. Prior to this, features for the points in $\boldsymbol{X}^{l+1}$ are computed, divided into two segments. The primary feature segment, sourced from $\boldsymbol{X}^{l+1}$, reflects the current point cloud's shape and guides its refinement. Extracted using an advanced PointNet++ (Qi et al., 2017) framework, these features are represented as $\boldsymbol{F}_1^{l+1} = \{f_{1,j}^{l+1} \in \mathbb{R}^{d_1^{l+1}} | 1 \leq j \leq N_{l+1}\}$. Conversely, the secondary feature segment originates from the preceding PU module's output $\boldsymbol{X}^l$, facilitating the encoded feature propagation through the PU layers, thus molding the final point cloud. Merging these with the first segment, using the feature transformation (FT) module, yields the consolidated features $\boldsymbol{F}^{l+1} = \{(f_{1,j}^{l+1}, f_{2,j}^{l+1}) \in \mathbb{R}^{d^{l+1}} | 1 \leq j \leq N_{l+1}\}$, where d^{l+1} is the total feature dimensionality.

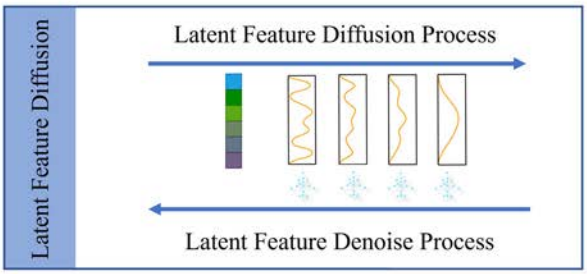
Through iterative application of PU and FT modules, the sparse latent points are progressively upsampled to a dense point cloud comprising 2048 points. The final PU module, besides facilitating upsampling, predicts normals for the reconstructed point cloud. The decoder initiates with sparse



(a) The autoencoder encodes a mesh to features at the sparse latent points and decodes it back to a mesh.

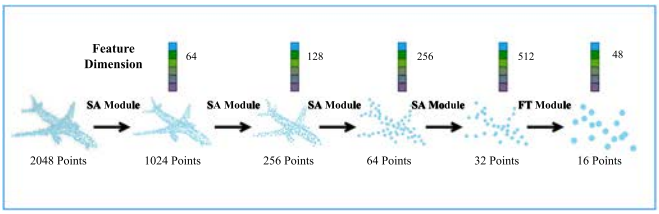


(b) The DDPM learns the distribution of the sparse latent points.

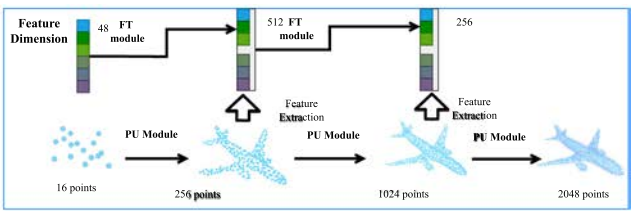


(c) The DDPM learns the distribution of features at latent points.

Fig. 4 Overview of geometry generation model with our sparse latent point diffusion model and the two latent diffusion models



(a) The point cloud encoder.



(b) The point cloud decoder.

Fig. 5 Architecture of the point cloud autoencoder

latent points $\boldsymbol{X}^1$ and their features $\boldsymbol{F}^1$, evolving to generate the reconstructed point cloud $\boldsymbol{X}^4$ with 2048 points and normals $\boldsymbol{F}^4$.

Training of the autoencoder The point cloud autoencoder is trained to encode the input point cloud and then reconstruct the point cloud. The input to the autoencoder is point cloud $\boldsymbol{X}_{\text{in}}$ (2048 points) with normals $\boldsymbol{F}_{\text{in}}$ sampled from the meshes in the dataset. The supervision is added on all the intermediate upsampling results in the point cloud decoder: $\boldsymbol{X}^2$, $\boldsymbol{X}^3$, $\boldsymbol{X}^4$. The loss is the sum of the Chamfer distance (CD) between $\boldsymbol{X}_{\text{in}}$ and $\boldsymbol{X}^2$, $\boldsymbol{X}^3$, $\boldsymbol{X}^4$, respectively. Note that when computing the CD loss between $\boldsymbol{X}_{\text{in}}$ and $\boldsymbol{X}^2$, $\boldsymbol{X}^3$, we first downsample $\boldsymbol{X}_{\text{in}}$ using farthest point sampling to the same number of points as $\boldsymbol{X}^2$ and $\boldsymbol{X}^3$, respectively. We also add a normal consistency loss between the ground-truth normals $\boldsymbol{F}_{\text{in}}$ and the predicted normals $\boldsymbol{F}^4$ with a weight of 0.1. See Appendix B.3 for details of this loss. We further add a slight Kullback–Leibler divergence loss (weight 10^{-5}) between the encoded features $\boldsymbol{F}^1$ and a standard normal distribution. This regularization term is to encourage the latent feature space to be simple and smooth, so that we can perform manipulation and interpolation in this space. Before the encoder encodes the input point cloud $\boldsymbol{X}_{\text{in}}$ to the sampled sparse latent points $\boldsymbol{X}^1$, we add a Gaussian noise with a standard deviation of 0.04 to the point positions in $\boldsymbol{X}^1$. This is to make the autoencoder more robust to the positions of the sparse latent points, so that even if the positions of the sparse latent points are not perfect (e.g., human-edited sparse latent points), the autoencoder can still well reconstruct the input point cloud.

4.1.2 DDPM Training in Sparse Latent Point Space

Post-training of the point cloud autoencoder, latent Denoising Diffusion Probabilistic Models (DDPMs) are trained within the autoencoder's latent space, with the autoencoder's parameters being frozen. Each point cloud is encoded to fea-

tures $\boldsymbol{F}^1 \in \mathbb{R}^{48}$ at sparse latent points $\boldsymbol{X}^1 \in \mathbb{R}^3$ for this purpose. Two distinct DDPMs are trained: the first targets the distribution of the sparse latent points $\boldsymbol{X}^1$, depicted in Fig. 4b, employing a lightweight PointNet++ (Qi et al., 2017) for the denoising function $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}^t, t)$ in Eq. 5.

The second DDPM, illustrated in Fig. 4c, is conditioned on the sparse latent points $\boldsymbol{X}^1$ and focuses on the feature distribution $\boldsymbol{F}^1$. For simplification, we represent the features $\boldsymbol{F}^1$ as a concatenated vector $\boldsymbol{f} \in \mathbb{R}^{48}$, and the sparse latent points $\boldsymbol{X}^1$ as $\boldsymbol{x} \in \mathbb{R}^3$. This conditional DDPM generates the feature vector $\boldsymbol{f}$ based on the sparse latent points $\boldsymbol{x}$. The diffusion process for this model modifies the variable $\boldsymbol{x}$ in Eq. 1 to $\boldsymbol{f}$, with the reverse process defined as:

$$p_{\boldsymbol{\phi}}(\boldsymbol{f}^0, \cdots, \boldsymbol{f}^{T-1} | \boldsymbol{f}^T, \boldsymbol{x}) = \prod_{t=1}^{T} p_{\boldsymbol{\phi}}(\boldsymbol{f}^{t-1} | \boldsymbol{f}^t, \boldsymbol{x}),$$
$$\text{where } p_{\boldsymbol{\phi}}(\boldsymbol{f}^{t-1} | \boldsymbol{f}^t, \boldsymbol{x}) = \mathcal{N}(\boldsymbol{f}^{t-1}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{f}^t, \boldsymbol{x}, t), \sigma_t^2 \boldsymbol{I}), \tag{6}$$

with the mean $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{f}^t, \boldsymbol{x}, t)$ parameterized as:

$$\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{f}^t, \boldsymbol{x}, t) = \frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{f}^t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\boldsymbol{f}^t, \boldsymbol{x}, t) \right). \tag{7}$$

The denoising network $\boldsymbol{\epsilon}_{\boldsymbol{\phi}}$ receives input concatenated from each feature in $\boldsymbol{f}^t$ and the corresponding point in $\boldsymbol{x}$, effectively treating the input as a sparse point cloud with noisy features. An enhanced PointNet++ (Qi et al., 2017) in PDR is utilized as the denoising network $\boldsymbol{\epsilon}_{\boldsymbol{\phi}}$. The training loss for this network is expressed as:

$$L(\boldsymbol{\phi}) = \mathbb{E}_{(\boldsymbol{X}_{\text{in}}, \boldsymbol{F}_{\text{in}}) \sim p_{\text{data}}} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\phi}} \left(\sqrt{\bar{\alpha}_t} \boldsymbol{f} + \sqrt{1-\bar{\alpha}_t} \boldsymbol{\epsilon}, \boldsymbol{x}, t \right) \right\|^2, \tag{8}$$

where $\boldsymbol{X}_{\text{in}}$ and $\boldsymbol{F}_{\text{in}}$ denote the point cloud and its normals from the dataset, respectively, and $\boldsymbol{\epsilon}$ is the Gaussian noise.

Appendix A.3 elaborates on the architecture of both DDPMs. These models enable both unconditional and controllable 3D shape generation. Unconditional generation cascades the two DDPMs: the first generates sparse latent points, followed by the second generating corresponding features, which are then decoded into a point cloud. Controllable generation involves adjusting the sparse latent points' positions, feeding these to the second DDPM to generate feasible features, and subsequently decoding these into a point cloud.

4.2 Texture Generation

After presenting mesh generation, we extend our SLIDE for subsequent texture generation with our generated geometry $\mathcal{M}$ and a text prompt $\mathbf{y}$ to improve our versatility.

Traditional approaches relying on a canonical sequence for camera views often result in inconsistencies, as depth-conditioned diffusion models trained on single-view 2D image priors struggle to fully capture the structure from a single viewpoint. SLIDE overcomes these challenges by integrating multi-view diffusion models with depth-conditioned diffusion models, ensuring coherent multi-view image generations that are congruent with the underlying mesh geometry. This is achieved through a two-phase texture generation method, consisting of an "*coarse phase*" and a "*refinement phase*", which collectively enhance the alignment and consistency of textures across different viewpoints. An overview of SLIDE's approach to texture generation is depicted in Fig. 3, with further details provided in subsequent sections.

4.2.1 Preliminary

Following the definition in Ho et al. (2020b) and Song et al. (2020), the multi-step generation process of diffusion models to reach the final result $\mathbf{z}_0$ is based on the sampling from the posterior defined as:

$$q\left(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \hat{\mathbf{z}}_{0|t}\right) = \mathcal{N}\left(\mathbf{z}_{t-1}; \tilde{\mu}_t\left(\mathbf{z}_t, \hat{\mathbf{z}}_{0|t}\right), \sigma_t \mathbf{I}\right), \tag{9}$$

where $\tilde{\mu}_t$ is a linear combination of $\mathbf{z}_t$ and $\hat{\mathbf{z}}_{0|t}$, and the sampling of $\mathbf{z}_{t-1}$ can be re-parameterized as

$$\mathbf{z}_{t-1} = \gamma_t \hat{\mathbf{z}}_{0|t} + \eta_t \mathbf{z}_t + \sigma_t \boldsymbol{\epsilon}_t, \tag{10}$$

with coefficients γ_t, η_t, and σ_t determined by the diffusion schedule, and $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. $\hat{\mathbf{z}}_{0|t}$ is an estimation of $\mathbf{z}_0$ at the current timestep t with a neural network $f_{\boldsymbol{\theta}}(\mathbf{z}_t, t)$. Since in the diffusion process $\mathbf{z}_t = \sqrt{\alpha_t}\mathbf{z}_0 + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, instead of directly predicting $\mathbf{z}_0$ given $\mathbf{z}_t$, we can equivalently predict $\hat{\boldsymbol{\epsilon}}_t = \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t, t)$ given $\mathbf{z}_t$ and let

$$\hat{\mathbf{z}}_{0|t} = f_{\boldsymbol{\theta}}(\mathbf{z}_t, t) = \frac{\mathbf{z}_t - \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t, t)}{\sqrt{\alpha_t}}, \tag{11}$$

where $\{\alpha_t\}_{t=1}^T$ are the parameters of the diffusion process related to the diffusion schedule.

4.2.2 Coarse Phase

In this section, we detail our approach for generating textures utilizing multi-view diffusion model priors, aimed at producing N rendered views $\{\mathbf{z}_{0,n}\}_{n=1}^N$ that exhibit 3D consistency from the initial noisy inputs $\{\mathbf{z}_{T,n}\}_{n=1}^N$. Our method seeks to iteratively enhance multi-view texture generation by incorporating depth information directly into the generative process. The specific selection of these two diffusion models is explained in a sec. 5.1.

Algorithm 1 Coarse phase

Input: Set of rendered views in latent space $\mathbf{z}_t$, timestep T, diffusion schedule parameter α_t, coefficients γ_t, η_t, β
Output: consistent views $\mathbf{z}_0$
for $t = T$ to 0 **do**
  $\hat{\mathbf{z}}_{0|t,\text{multi-view}} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{z}_t - \sqrt{1-\alpha_t} \boldsymbol{\epsilon}_{\boldsymbol{\theta},\text{multi-view}} (\mathbf{z}_t, \mathbf{y}) \right)$
  $\hat{\mathbf{z}}_{0|t,\text{depth}} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{z}_t - \sqrt{1-\alpha_t} \boldsymbol{\epsilon}_{\boldsymbol{\theta},\text{depth}} (\mathbf{z}_t, \mathbf{y}) \right)$
  $\hat{\mathbf{z}}_{0|t,\text{compose}} = \beta \cdot \hat{\mathbf{z}}_{0|t,\text{depth}} + (1-\beta) \cdot \hat{\mathbf{z}}_{0|t,\text{multi-view}}$
  $\mathbf{z}_{t-1} = \gamma_t \hat{\mathbf{z}}_{0|t,\text{compose}} + \eta_t \mathbf{z}_t + \sigma_t \boldsymbol{\epsilon}_t$
end for
return $\mathbf{z}_0$

Employing the multi-step generation framework as defined in Eq. 10, our solution involves leveraging a blend of two distinct model outputs. The first focuses on enhancing images with depth conditions, while the second generates multi-view images. This integration is formalized as:

$$\hat{\mathbf{z}}_{t|0} = \beta \cdot \hat{\mathbf{z}}_{t|0,\text{depth}} + (1-\beta) \cdot \hat{\mathbf{z}}_{t|0,\text{multi-view}}, \tag{12}$$

where the terms $\hat{\mathbf{z}}_{t|0,\text{depth}}$ and $\hat{\mathbf{z}}_{t|0,\text{multi-view}}$ adhere to the formulation in Eq. 11. Here, $\boldsymbol{\epsilon}_{\boldsymbol{\theta},\text{depth}}(\cdot)$ and $\boldsymbol{\epsilon}_{\boldsymbol{\theta},\text{multi-view}}(\cdot)$ represent the models predicting noise based on text prompts $\mathbf{y}$. The coefficient $\beta \in [0, 1]$ serves as a hyper-parameter to balance depth and shape information, typically set above 0.5 during the early stages and gradually adjusted.

The initial texture generation phase leverages single-view and multi-view priors to create a preliminary texture map, though it faces challenges of incomplete coverage and limited resolution. A refinement phase with a binary refinement algorithm addresses these issues, focusing on enhancing texture detail and overall coverage, thereby improving the realism and fidelity of the generated textures.

In the coarse phase of our method, although we may not be able to generate textures for every area and there are regions lacking textures that need to be addressed in the following phase, the generated textures exhibit significant consistency across various viewpoints for most areas. This can be observed in Appendix B.12, showing that texture consistency can be attained during the coarse phase. The strength of the multi-view consistency of our method lies in its use of a multi-view diffusion model that generates textures across multiple viewpoints simultaneously. This way significantly mitigates the inconsistencies that typically arise from single-viewpoint texture generation, thus enhancing overall texture coherence. However, it is important to note that even with a multi-view diffusion approach, achieving 100% consistency across all viewpoints is challenging. The model may still produce inconsistencies under certain complex conditions. In the current benchmark, the level of consistency achieved is relatively satisfactory and fulfills a substantial improvement over single-viewpoint methods.

Algorithm 2 Refinement phase

Input: Preliminary texture map $\mathbf{Z}_t$, number of denoising steps T, rendering process R, depth maps $\{\mathbf{D}_n\}_{n=1}^N$, Inpainting Diffusion Model $\mathcal{I}$, Depth-conditioned Diffusion Model $\mathcal{D}$
Output: Enhanced texture map $\mathbf{Z}_0$
for $t = T$ to 0 **do**
  $(\mathcal{R}, \mathcal{G}) = \text{Segment}(\mathbf{Z}_t)$
  for $n = 1$ **to** N **do**
    $(\mathbf{z}_{t,n}, \mathbf{D}_n) = R(\mathbf{Z}_t)$
    if generate region $\mathcal{G}$ exists in $\mathbf{z}_{t,n}$ **then**
      $\mathbf{z}_{\mathcal{G},n} = \mathcal{I}(\mathbf{z}_{t,n}, \mathbf{D}_n)$
    end if
    $\mathbf{z}_{\mathcal{R},n} = \mathcal{D}(\mathbf{z}_{t,n}, \mathbf{D}_n)$
  end for
  $\mathbf{Z}_t = \text{Aggregate}\left(\{\mathbf{z}_{\mathcal{R},n}, \mathbf{z}_{\mathcal{G},n}\}_{n=1}^N\right)$
end for
return $\mathbf{Z}_0$

4.2.3 Refinement Phase

To overcome the challenges of low resolution and untextured areas found in the coarse phase, we proposed a refinement strategy. This approach initially segments the texture map produced earlier into two regions: the "refine region" for areas previously textured, and the "generate region" for those that remained untextured. This segmentation facilitates targeted texture enhancements tailored to each zone's specific needs.

The refinement process begins with rendering at specific angles to acquire texture and depth maps. Subsequently, noise is introduced to the texture map, followed by the application of advanced denoising techniques. During the denoising phase, for "generated regions", an inpainting stable diffusion model is preferred over the traditional depth-conditioned model for certain denoising steps. This strategy enables direct texturing of generated areas through inpainting, ensuring their seamless integration with surrounding textures. In "refined regions", depth images are utilized as conditional inputs to improve depth-sensitive texturing.

The refinement phase ends with a detailed texture map that keeps the uniform appearance from earlier phases and covers areas that were previously untextured, greatly improving the resolution. This results in better quality, complete coverage, high fidelity, and more realistic textures.

5 Experiments

In this section, the evaluation is conducted as three comprehensive parts, aimed at showing the effectiveness on textured mesh generation, geometry generation, and texture generation against extensive existing methods.

Fig. 6 The comparison between GET3D and ours. All the textured mesh including geometry and texture are generated by our method and baseline

5.1 Implementation Details

We train the components of our mesh generative model using the ShapeNet dataset (Chang et al., 2015), as preprocessed by (Peng et al., 2021a), which comprises 13 object categories and splits into training and validation sets. Regarding the autoencoder component, it is trained with Adam at a learning rate of 0.001 and batch size of 32 for 6000 epochs, on five ShapeNet categories (Airplane, Cabinet, Car, Chair, and Lamp), selecting checkpoints based on the lowest reconstruction error for subsequent latent DDPM training. Subsequently, two latent DDPMs are trained in the fixed autoencoder's latent space to learn distributions of sparse latent points' positions and features, both over 10000 epochs with specific EMA rates for different categories. For more training details and settings for all baselines, please refer to Appendix B.1 and Appendix B.2.

For the texture generation, at its coarse phase, we initially focus on four distinct views—front, left, back, and right. During this phase, we employ two stable diffusion models: a depth-conditioned diffusion model and a multi-view diffusion model. Specifically, for the depth-conditioned model, we utilize Stable Diffusion V1-5 coupled with the Depth ControlNet module v1-1 from the Huggingface Diffusers library. For multi-view texture generation, we use MVDream (Shi et al., 2023), which efficiently generates concurrent textures for the initial views. The refinement phase broadens our perspective to ten camera positions, encircling the object at 45-degree intervals at the equator, with two additional elevated views at 30 degrees, enhancing texture coverage comprehensively. We employ a denoising timestep of 50 to achieve a fully textured map.

5.2 Evaluation Metrics

To evaluate the final textured meshes, the Fréchet Inception Distance (FID) and Kernel Inception Distance (KID) metrics are utilized. The process entails rendering the textured meshes produced by our methodology and those from established baselines, alongside the ground truth textured meshes sourced from the dataset. Subsequently, FID and KID calculations are performed across these rendered image sets, facilitating a quantified analysis of the congruence between the generated and ground truth textures.

For the structural assessment, point clouds comprising 2048 points along with their normals are uniformly sampled from both generated meshes and reference meshes within the validation set. We use established point cloud metrics, including 1-Nearest Neighbor (1-NN), Minimum Matching Distance (MMD), and Coverage (COV). These metrics necessitate a distance metric, for which the Chamfer distance (CD) and Earth Mover's distance (EMD) are commonly utilized to quantify the disparity between two point clouds. To enhance the evaluation granularity, we incorporate the normal consistency loss, which offers a nuanced reflection of the surface curvature differences between the compared meshes.

While 1NN, MMD, and COV are widely recognized within the 3D generation community, they have their limitations. For instance, PointFlow (Yang et al., 2019) highlights shortcomings in the COV and MMD metrics for assessing the quality of generated point clouds, noting that these metrics may not accurately capture the distances and similarities between generated and real samples. To address these issues, PointFlow introduces the 1NN metric for improved similarity evaluation. However, despite its advantages, the 1NN metric also faces challenges, particularly in effectively capturing the relationship between the generated distribution and the training distribution, as emphasized by SDF-StyleGAN (Zheng et al., 2022). To further substantiate the quality of our generated outputs, we also conduct a user study. More details on this loss are delineated in Appendix B.3.

For the texture assessment, we use a rendering-based method where textures from different generation methods are rendered as images. Discrepancies between these images and ground-truth textures are assessed using FID, KID, and CLIP score. Beside, we perform a user study that collects the

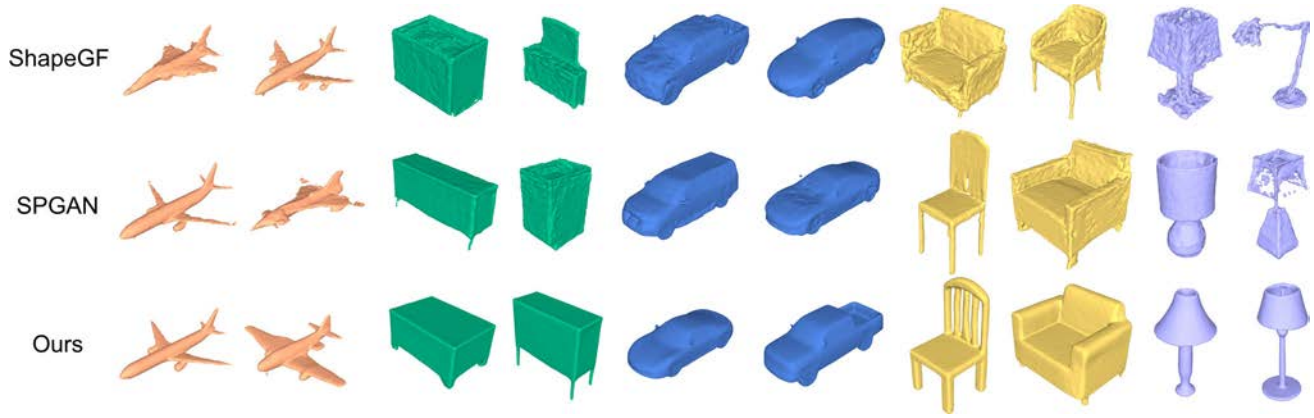


Fig. 7 Meshes generated by SLIDE and baselines. We can see that meshes generated by our method are more visually appealing. More examples of other baselines and SLIDE are provided in Appendix B.4

Table 1 User study results of textured mesh

Metric	GET3D
Geometry quality	52.57%
Alignment of geometry and texture	56.57%
Texture quality	84.00%

This table shows the proportion of participants in a user study who favored our method over the baseline, based on 175 real-world human responses. The percentages reflect how many users, out of the total number of participants, found our method to produce better results compared to the baseline

human subjective feedback to assess the prompt alignment and consistency of generation.

5.3 Textured Mesh Generation

In this section, we evaluate the textured meshes generated by our method, comparing them against baseline methods capable of generating both structure and texture simultaneously. Due to the unavailability of the 3DGEN (Gupta et al., 2023) code, we exclusively utilize GET3D (Gao et al., 2022) as our baseline for comparison, which is known for its ability to generate textured meshes.

Qualitative Comparison Figure 6 presents a comparative analysis of our results against those of GET3D. The visual assessment indicates that our method produces more vivid renderings compared to GET3D. This distinction is particularly noticeable given that the textures in ShapeNet are predominantly simple and white, leading to GET3D often producing monochromatic samples. Furthermore, the texture edges in GET3D's outputs lack sharpness due to their texture field generation approach, which exhibits limited sensitivity to coordinate changes.

Quantitative Comparison Table 1 presents the user study comparing our rotating GIF results with those of GET3D.

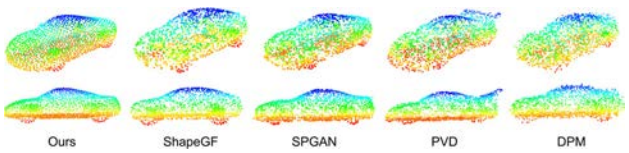


Fig. 8 Point clouds generated by our method and baselines. More examples are provided in Appendix B.5

The data reveals that user preferences for geometry quality and the alignment of geometry and texture are comparable, suggesting that our generated samples do not surpass GET3D in terms of geometric fidelity. However, a majority of respondents favored our method over GET3D in terms of texture quality, indicating that SLIDE outperforms in rendering textures on generated meshes.

5.4 Geometry Generation

In Fig. 7, we present some mesh exemplars generated by SLIDE and baselines (see Appendix B.6, B.7, and B.8 for more examples). We can observe that SLIDE generates meshes of the highest visual quality, with smooth surfaces and sharp details. Since all meshes are reconstructed from the generated point clouds using the same method SAP, this means the quality of the generated point clouds greatly affects the quality of the reconstructed meshes. To verify this, we provide an example of generated point clouds in Fig. 8 (see more examples in Appendix B.7). Indeed, we can see that point clouds generated by SLIDE spread more uniformly on the surface of the objects, and bear less noise compared with other methods. We attribute this to the design of our novel point cloud autoencoder.

Quantitatively, we compute 1-NN and COV metrics for the reconstructed meshes, respectively presented in Table 2 and in Table 3 (MMD result is in Appendix B.4), and the results of the user study are shown in Table 4. Our method achieves

Table 2 1NN-Acc (Percentage) comparison of meshes generated by our method and baselines

	Airplane			Cabinet			Car			Chair			Lamp		
	CD	EMD	N.C	CD	EMD	N.C	CD	EMD	N.C	CD	EMD	N.C	CD	EMD	N.C
TreeGan	81.31	71.78	84.65	69.75	74.20	55.73	91.32	75.23	87.18	70.24	62.04	56.94	74.68	59.74	**50.43**
ShapeGF	73.39	71.29	76.36	61.46	**53.82**	56.37	**67.29**	**59.01**	54.47	56.57	54.14	54.65	57.79	**51.30**	54.33
PVD	88.86	83.91	86.14	60.82	65.28	58.60	70.09	59.87	58.14	**55.39**	52.36	55.10	58.44	56.06	76.19
DPM	75.50	68.81	**50.37**	62.42	63.69	**49.68**	86.38	79.44	**50.00**	66.17	69.50	**49.93**	66.23	61.04	52.60
SPGAN	82.05	70.42	83.29	70.38	61.15	60.19	85.18	75.23	59.61	79.03	75.85	77.99	67.97	64.07	70.13
Ours (Centroid)	**70.17**	**65.84**	72.40	55.73	59.24	55.41	69.09	64.62	53.40	56.72	**51.18**	53.77	58.44	58.87	52.81
Ours (Random)	72.40	67.82	72.52	**53.50**	56.69	54.78	70.09	63.68	53.34	56.57	53.10	53.91	**56.28**	54.11	54.33

Bold values indicate the best performance
"N.C." denotes normal consistency loss. "Centroid" denotes the FPS method in which we choose the centroid of the point cloud as the initial point, and "Random" denotes the FPS method in which we randomly choose a point as the initial point

Table 3 COV (Percentage) comparison of meshes generated by our method and baselines

	Airplane			Cabinet			Car			Chair			Lamp		
	CD	EMD	N.C	CD	EMD	N.C	CD	EMD	N.C	CD	EMD	N.C	CD	EMD	N.C
TreeGan	46.29	43.81	31.68	43.31	**50.96**	29.94	33.24	34.18	8.28	49.63	48.89	26.29	47.19	46.75	29.87
ShapeGF	**49.50**	41.58	**41.34**	45.86	49.04	38.85	**44.73**	**47.80**	14.82	**52.88**	**50.96**	34.12	50.65	**55.84**	**36.80**
PVD	34.41	34.9	32.67	45.85	48.41	30.57	40.05	41.39	**30.68**	48.15	50.52	32.20	51.08	55.41	36.79
DPM	43.32	**48.76**	32.18	49.68	50.32	33.76	33.24	35.11	7.21	44.02	47.27	26.44	48.48	52.38	29.44
SPGAN	41.09	46.29	34.65	39.49	42.68	32.48	32.58	36.72	12.68	30.13	31.91	21.86	46.75	49.78	34.63
Ours (Centroid)	48.51	46.29	39.85	**52.87**	47.77	37.58	38.99	39.92	12.55	49.19	49.63	34.71	**52.81**	52.38	36.36
Ours (Random)	48.76	47.03	37.87	50.96	**50.96**	**40.76**	43.12	41.26	13.89	48.89	50.37	**34.86**	51.08	52.81	33.77

Bold values indicate the best performance
"N.C." denotes normal consistency loss

Table 4 User study comparison results

Method	TreeGan	ShapeGF	PVD	DPM	SPGAN
Geometry quality	91.0%	85.0%	89.0%	88.0%	92.0%

The study involved 500 responses from 25 participants, who compared two sets of geometric generation results: one produced by our method and the other by baseline methods. The percentages in the table reflect the proportion of responses favoring our method over the baselines

superior performance on several categories. In terms of efficiency, the average generation time for a single point cloud of SLIDE is about 0.2s (See Appendix B.10 for more details of the generation time.) tested on a single NVIDIA A100 GPU, while the DDPM-based method that directly trains generative models on dense point clouds, PVD (Zhou et al., 2021b), need 2.93 s to generate a point cloud tested on the same A100 GPU. LION (Zeng et al., 2022b) reports it needs 27.12s per shape. This proves the superior efficiency of SLIDE over previous methods.

Controllable Generation As mentioned in Sect. 4.1.2, we can use the sparse latent points to control the generated mesh. Figure 2 shows that we can flexibly control the overall scale of the generated mesh as well as change the position, scale, or shape of a part of the mesh. It is worth noting that we achieve this without any part annotations of the dataset. SLIDE is also able to generate diverse meshes even for the same set of sparse latent points, which we use two pairs of examples to illustrate in Fig. 9.

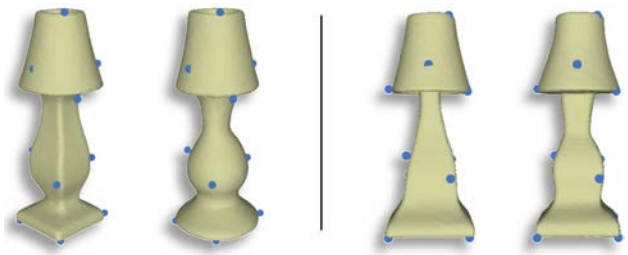


Fig. 9 SLIDE is able to generate diverse meshes for the same set of sparse latent points due to the stochasticity in the feature generation process. Here are two pairs of generated lamps for the same set of latent points

Table 5 Quantitative comparisons on Objaverse subset and ShapeNet Car including FID and KID scores, and texture generation time for a single object across different methods

Metrics	Dataset	Latent-paint	Text2Tex	TEXTure	SyncMVD	Ours
FID ↓	Objaverse	58.47	61.08	40.68	39.78	**35.36**
KID($\times 10^{-3}$) ↓	Objaverse	18.23	23.21	7.70	6.97	**4.54**
Clip-Score ↑	Objaverse	77.84%	76.25%	81.12%	82.87%	**83.10%**
FID ↓	ShapeNet Car	88.64	72.87	75.14	68.92	**62.67**
KID($\times 10^{-3}$)↓	ShapeNet Car	22.31	21.14	13.65	12.35	**9.05**
Clip-Score ↑	ShapeNet car	64.63%	63.28%	67.16%	68.39%	**69.10%**
Generation time ↓		3.6 min	8 min	2.9 min	**2 min**	3.2 min

Bold values indicate the best performance
The FID and KID score is calculated on the set of images synthesized by the original Stable Diffusion with ControlNet

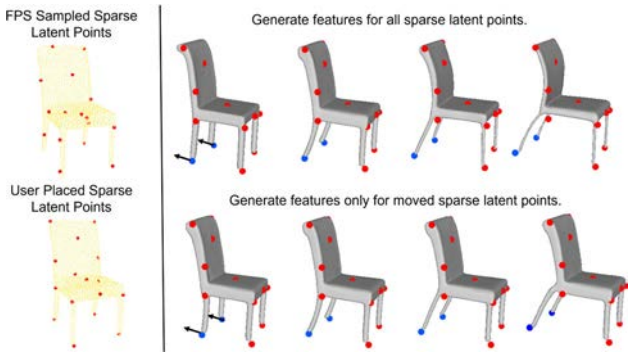


Fig. 10 Use manually placed sparse latent points to control the rear legs of the generated chairs. The blue points are moving and the red points are fixed. The top row generates new features for all sparse latent points, and the bottom row generates new features only for moved points and fixes the features of the rest points (Color figure online)

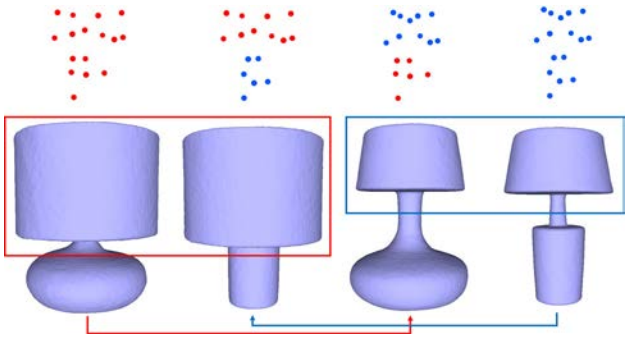


Fig. 12 Perform shape combination. The first row is the sparse latent points of the original two lamps and the combined sparse latent points. The second row are the original two lamps (two sides) and the two lamps (middle two) obtained by combining the top part and bottom part of the original lamps

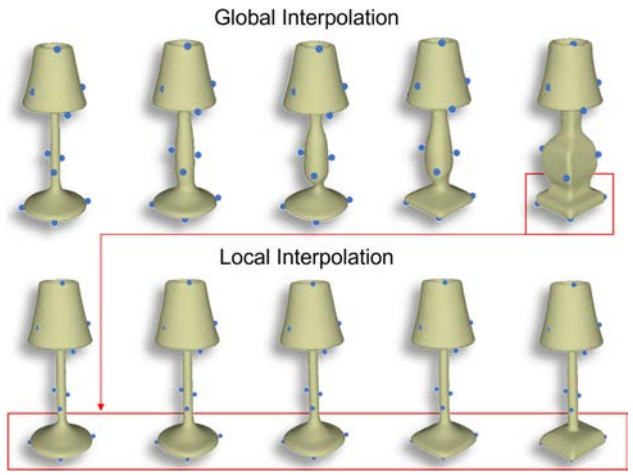


Fig. 11 SLIDE is able to perform both global and local interpolations. The first row is an example of global interpolation. The second row interpolates between the bottom of the two lamps

Besides, the sparse latent points in Fig. 2 are obtained by FPS. However, we can also manually place the sparse latent points at regions of interest other than FPS sampled points and control the corresponding part. Figure 10 gives an example where we manually select the sparse latent points and control the rear legs of a chair. When a user manipulates the sparse latent points, we typically utilize feature DDPM to generate new features for the relocated points. This procedure includes two distinct methodologies for feature generation. The first, depicted in the top row of the figure, involves exclusively generating new features for the manipulated points. This approach maintains the local geometry of the remaining points intact. The second method, represented in the bottom row, involves regenerating features for all points following their displacement, leading to alterations in the object's overall geometry. Furthermore, the figure on the left demonstrates two disparate strategies for point placement: the Farthest Point Sampling (FPS) method and a user-specified point placement approach. More contents are described in Appendix A.5.

Shape Interpolation & Shape Combination Our model can implement shape interpolation by interpolating both the positions and features between the corresponding latent points of the two shapes (see Appendix B.11 for more details). In Fig. 11, we give examples of both global interpolation and local interpolation. The top row of the figure represents global interpolation, where the middle lamp is the result of interpolating between the lamps on the far left and far right, encompassing both the lampshade and the lamp base.

Table 6 Percentage value of users who prefer our method with the baselines in 700 responses

Metric	Latent-paint (%)	TEXTure (%)	Text2tex (%)	SyncMVD (%)
Overall quality	97.14	60.57	73.14	62.85
Align with prompt	92.00	62.28	68.57	74.85
Texture consistency	94.85	64.57	74.85	78.85

Fig. 13 Qualitative comparisons on Objaverse among SLIDE, Latent-Paint (Metzer et al., 2023), Text2Tex (Chen et al., 2023a), TEXTure (Richardson et al., 2023) and SyncMVD (Liu et al., 2023a)

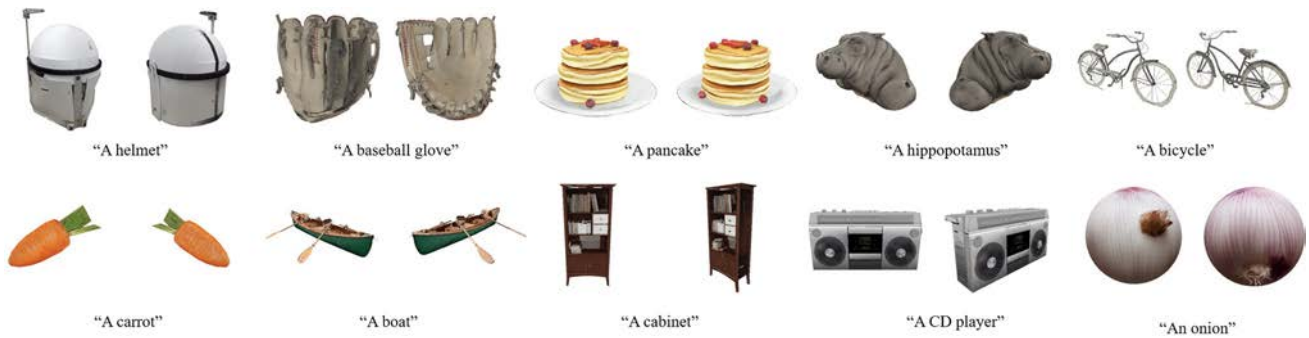


Fig. 14 More results generated by our method, using geometry from Objaverse, are presented

Fig. 15 Texture generated with different prompts of car and briefcase. It showcases SLIDE's capacity to render realistic wood grain and aging effects, proving attention to detail and material specificity

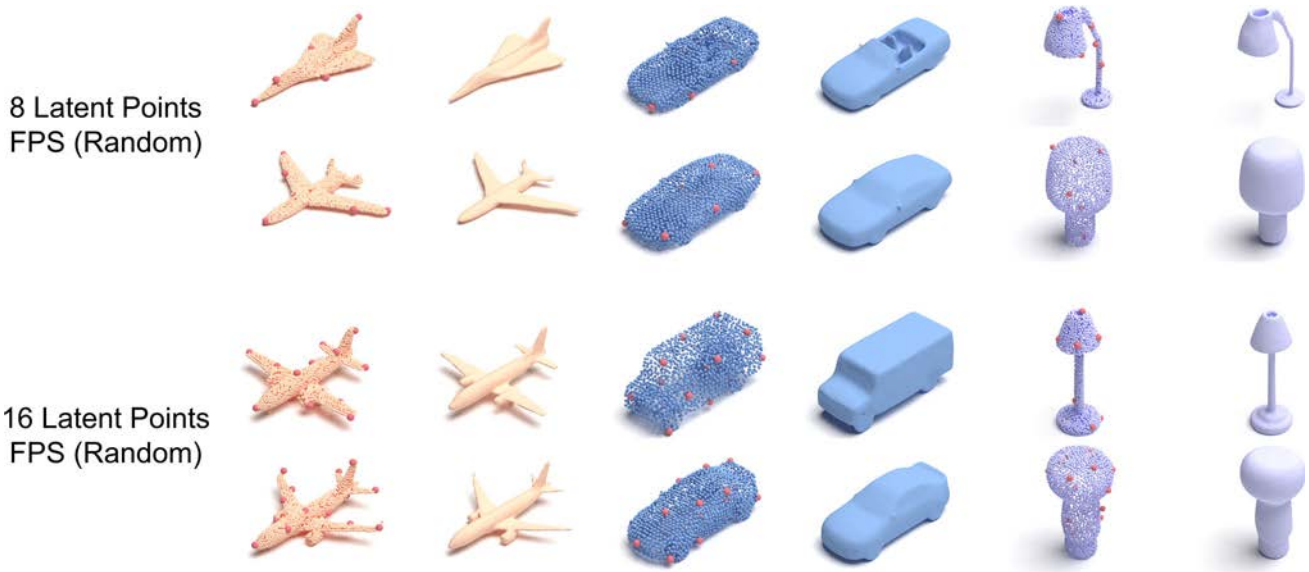


Fig. 16 An ablation study was conducted on the number of latent points (8, 16, 32) and the sampling method (FPS or random sampling), showcasing meshes and point clouds generated by our method under varying settings. Note that only results for 8 and 16 points are presented here, while the full set of results is provided in Appendix B.10

The bottom row, on the other hand, showcases local interpolation. It demonstrates the result of interpolating solely the lamp base while keeping the lampshade constant. The lamps in the middle are the results of local interpolation between the distinct lamp bases of the lamps on the far left and far right. Similarly, we can also perform shape combinations by simply combining the sparse latent points and their features from two or more source shapes to form new shapes. This is proved by the illustration in Fig. 12. These examples show the potential of SLIDE in the diverse extensions.

5.5 Texture Generation

Quantitative Comparisons Considering the the availability of source codes, we choose Latent-Paint, TEXTure, Text2Tex for the comparison. We utilize the official codes of Paint, TEXTure, Text2Tex to produce quantitative results, and evaluate these methods on a subset of textured meshes from the Objaverse dataset as (Chen et al., 2023a). We render the depth map of all meshes in 8 fixed canonical views at a resolution of 512×512, and condition depth ControlNet on both these depth maps. Regarding the textual prompt, the template "A ⟨category⟩" is used for all models. The results of each method in terms of the FID & KID score are summarized in Tab. 5, which reveal that the textures generated by SLIDE closely resemble the ground truth set in terms of appearance, surpassing other methods.

Moreover, a user study is conducted to analyze the quality and fidelity of the synthesized textures based on human judgment. For each comparison between SLIDE and baselines, participants were randomly shown 28 pairs of renders, one from the baselines and one from our method. Participants

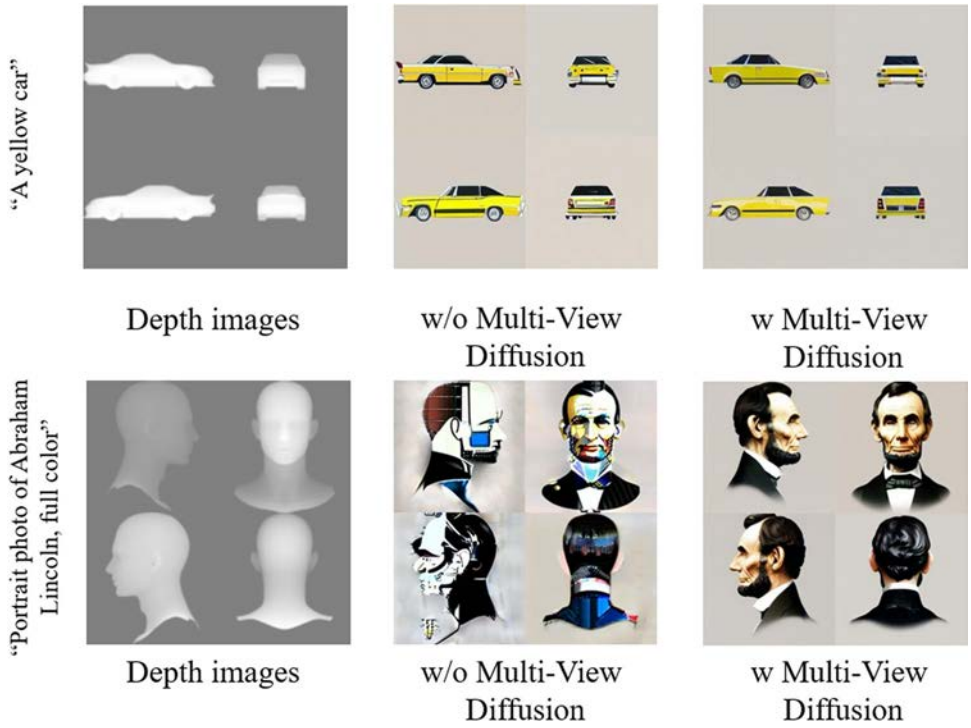


Fig. 17 Ablation study of introducing Multi-View Diffusion Model in the coarse phase. The results are multi-views conditioned on the depth images in first generate stage

Table 7 Ablation study on the number of latent points (8,16,32) and the method to sample them (FPS or random sampling)

Number of	Sampling	Airplane		Car		Lamp	
Latent points	Method	Error	Time	Error	Time	Error	Time
8	FPS(Random)	0.94	**0.137**	2.22	**0.194**	2.18	**0.141**
16	FPS(Random)	0.81	0.197	2.15	0.242	1.70	0.196
32	FPS(Random)	**0.74**	0.414	**2.10**	0.450	**1.47**	0.414
16	Random	0.86	0.197	2.12	0.235	1.75	0.204

Bold values indicate the best performance
We report the reconstruction error (CD $\times 10^{-3}$) of the autoencoder and the average generation time (s) per sample

Table 8 Ablation study on the number of latent points (8,16,32) and the method to sample them (FPS or random sampling)

Data	Number of	Sampling	Generation	Airplane			Car			Lamp		
Format	Latent Points	Method	Time	CD	EMD	N.C	CD	EMD	N.C	CD	EMD	N.C
Mesh	8	FPS(Random)	0.21	71.91	63.74	72.03	**68.62**	67.42	**53.00**	64.72	55.63	58.01
	16	FPS(Random)	0.26	**70.17**	67.08	**71.66**	70.09	**63.68**	53.34	**56.93**	53.90	**53.25**
	32	FPS(Random)	0.49	72.28	**62.13**	73.51	71.76	66.15	53.40	62.99	54.98	56.06
	16	Random	0.28	71.04	64.85	73.51	68.76	64.29	**53.00**	60.61	**52.81**	56.49
	8	FPS(Random)	0.19	61.14	**63.37**	–	57.21	67.49	–	56.06	58.23	–
Point	16	FPS(Random)	0.57	64.36	75.74	–	58.28	**64.22**	–	**53.25**	56.49	–
Cloud	32	FPS(Random)	0.47	68.07	66.96	–	60.08	66.15	–	54.76	53.25	–
	16	Random	0.26	**57.30**	65.72	–	**56.74**	65.89	–	54.33	**52.60**	–

Bold values indicate the best performance
We report 1NN-Acc (Percentage) of meshes and point clouds generated by our method under different settings and the average generation time (seconds) per sample. "N.C." denotes normal consistency loss

Table 9 Ablation study on the number of latent points (8,16,32) and the method to sample them (FPS or random sampling). We report MMD of meshes and point clouds generated by our method under different settings and the average generation time (seconds) per sample

Data Format	Number of Latent Points	Sampling Method	Generation Time	Airplane			Car			Lamp		
				CD	EMD	N.C	CD	EMD	N.C	CD	EMD	N.C
Mesh	8	FPS(Random)	0.21	4.46	3.76	**3.02**	**4.50**	3.23	**3.21**	**22.48**	**11.29**	4.34
	16	FPS(Random)	0.26	**4.41**	3.90	3.06	4.58	3.18	3.23	22.59	11.31	**4.31**
	32	FPS(Random)	0.49	4.50	**3.75**	3.14	4.64	**3.14**	3.25	23.07	11.49	4.44
	16	Random	0.28	4.49	3.76	3.08	4.52	3.19	**3.21**	23.63	11.68	4.37
	8	FPS(Random)	0.19	4.14	**3.63**	–	**3.95**	3.13	–	21.55	11.24	–
Point	16	FPS(Random)	0.57	4.05	4.09	–	4.07	**2.99**	–	20.34	**11.15**	–
Cloud	32	FPS(Random)	0.47	4.30	3.80	–	4.15	3.14	–	21.70	11.55	–
	16	Random	0.26	**4.01**	3.70	–	4.04	3.08	–	**19.61**	11.16	–

Bold values indicate the best performance
CD, EMD, and normal consistency (N.C.) losses are multiplied by 1000, 100, and 10, respectively

Table 10 Ablation study on the number of latent points (8,16,32) and the method to sample them (FPS or random sampling)

Data Format	Number of Latent points	Sampling Method	Generation time Time	Airplane			Car			Lamp		
				CD	EMD	N.C	CD	EMD	N.C	CD	EMD	N.C
Mesh	8	FPS (Random)	0.21	45.05	44.31	39.85	42.19	41.92	**15.35**	49.35	51.08	35.06
	16	FPS (Random)	0.26	45.54	42.08	37.62	43.12	41.26	13.89	50.22	**59.74**	38.53
	32	FPS (Random)	0.49	47.03	44.80	**40.10**	43.12	**43.12**	12.28	**52.38**	55.41	34.63
	16	Random	0.28	**47.52**	**47.03**	39.60	**43.26**	42.06	13.08	48.48	53.25	**38.96**
	8	FPS (Random)	0.19	46.53	**46.53**	–	43.66	40.05	–	51.52	**56.28**	–
Point	16	FPS (Random)	0.57	49.50	37.62	–	**46.19**	41.66	–	50.65	53.68	–
Cloud	32	FPS (Random)	0.47	49.26	42.82	–	45.13	**43.12**	–	53.25	55.41	–
	16	Random	0.26	**50.74**	45.05	–	43.79	41.39	–	**54.11**	53.25	–

Bold values indicate the best performance
We report COV (Percentage) of meshes and point clouds generated by our method under different settings and the average generation time (seconds) per sample. "N.C." denotes normal consistency loss

evaluated our textures based on Overall Quality, Alignment with Textual Prompts, and Texture Consistency by choosing between two rendered images that best met these criteria. Details of the survey, which was distributed online and collected 700 responses from 25 users, are in Appendix B.13. The participant pool was approximately 60% male and 40% female, predominantly aged 20–40 years, with 60% from a computer science background and 40% from non-technical fields, providing a diverse range of feedback.

The results of the user study are summarized in Tab. 6. Compared to Latent-Paint, our method is distinctly favored by users, achieving a 97.14% preference rate. Additionally, a majority of users (60.57% and 73.14%) prefers our method over TEXTure and Text2Tex, highlighting its effectiveness in producing more consistent and natural results that align with human preferences.

Texture Generation Time In Table 5, we present a runtime comparison of the SLIDE model against baseline methods on a workstation equipped with a single NVIDIA A100 GPU. The SLIDE model's coarse phase is completed in 0.7 min, and its refinement phase in 2.5 min. This performance underscores the model's efficient computational design, facilitating high-quality texture generation suitable for rapid prototyping and real-time applications.

Qualitative Comparisons In the comparison, we randomly sample some geometric objects on Objaverse (Deitke et al., 2022) and show their corresponding texture generation results of SLIDE against the state-of-the-art texture generation methods, Latent-Paint (Metzer et al., 2023), Text2Tex (Chen et al., 2023a), TEXTure (Richardson et al., 2023) and SyncMVD (Liu et al., 2023a) in Fig. 13. From the figure, we can see that SLIDE excels over SDS methods like Latent-Paint, avoiding elevated color saturation, and performing better than progressive inpainting methods like Texture and Text2Tex, offering greater consistency and reduced artifacts. We provide more visualization in Fig. 14

The fundamental characteristic of our SLIDE model is its proficiency in text-driven controlled generation, which allows for the creation of diverse textures on the same geometric model based on various textual prompts. To showcase

this, we applied SLIDE to a single geometric model using various prompts, resulting in distinct textures that exemplify the model's ability to produce visually diverse outcomes from the same geometric form. The images in Fig. 15 illustrate this by displaying the geometric model textured with different results derived from unique prompts. This demonstrates the model's capacity to interpret and respond to natural language inputs, generating customized textures that accurately reflect given descriptions, affirming its potential in computational design and digital art applications.

5.6 Ablation Studies

In this section, we present the extensive ablation studies on different components of our SLIDE. Some detailed analysis e.g., the Set Abstraction (SAP) can be found in Appendix B.9.

5.6.1 Number of Sparse Latent Points

In this section, we conduct an ablation study on the number of sparse latent points (8,16,32) w.r.t. FPS, random sampling and our method. We report the autoencoder's reconstruction error and the latent DDPMs' average generation time of a single point cloud in Table 7. We also report 1-NN, MMD, and COV for point clouds and meshes under different settings in Appendix B.10. According to the results, we can see that increasing the number of latent points reduces the reconstruction error, but slows down the generation speed.

As for the generation quality reflected by the quantitative metrics in Table 8, Table 9, and Table 10, we find that the number of sparse latent points and the method to obtain them do not have a clear or significant impact. We thus further add a visual comparison of these settings in Fig. 16. As can be seen, these settings indeed generate point clouds and meshes of similar quality. The number of sparse latent points mainly affects the ability to perform controllable generation of our method. When we have too few sparse latent points, it restricts the complexity of the controls we can perform on an object. When we have too many sparse latent points, we can perform more delicate controls of the generated shapes, but it also becomes more complex and less intuitive to manipulate the sparse latent points since we need to consider their relative positions and ensure that they form a plausible skeleton of an object.

5.6.2 Number of Views in Refinement Phase

In this section, we conducted experiments to study how varying the number of views (N) affects the consistency and quality of the generated textures. As shown in the following Table 11, initially, increasing the number of views results in a significant improvement in texture quality. This enhancement is due to the ability to generate more detailed textural information across different angles, which effectively reduces ungenerated regions and common issues like blurry artifacts. However, as the number of views continues to increase beyond a certain point, the rate of improvement in texture quality diminishes. This plateauing effect occurs because additional views introduce more seams, which can complicate the texture consistency across different viewpoints. While a moderate increase in the number of views is beneficial, excessively high numbers do not correspond to significant gains in quality and can even negatively impact the overall texture consistency due to the increased complexity at the seams.

Table 11 We quantitatively study the effect of selecting different number of viewpoints in the refinement stage

Number of views	0	5	10	15	20
↓ FID	47.58	42.24	36.39	35.34	**35.15**
↓ KID ($\times 10^{-3}$)	14.36	10.37	5.65	5.24	**4.92**

Bold values indicate the best performance
Refining the results generated by coarse phase with more viewpoints improves the texture quality

5.6.3 Multi-view Diffusion Priors

In Fig. 17, we show that the Multi-View Diffusion priors can significantly influence the generation process of multi-view images at the initial stage, which confirms the benefit of a multi-view diffusion model markedly to the overall performance. This effectively addresses the limitations observed with the depth-conditioned diffusion model at a latent resolution of 32x32, underscoring the value of incorporating multi-view diffusion priors in achieving superior texture mapping results. The two situations listed here happen to be two specific cases of β values, with further ablation analyses regarding beta presented in Appendix B.15 (Fig. 18).

5.6.4 The Text Prompt

We have performed ablation experiments to examine the impact of text prompts on the quality of generated textures. These results are illustrated in the following Fig. 19. Without text prompts, our model still utilizes a depth-conditioned diffusion model that leverages the geometry of the object to guide texture generation. However, as shown in the figure, incorporating text prompts significantly enhances the texture quality and the alignment with the object's characteristics.

The text prompts provide precise control over the depth-conditioned and multi-view diffusion processes, enabling more accurate texture generation that aligns with our desired outcomes. This experiment demonstrates the effectiveness of

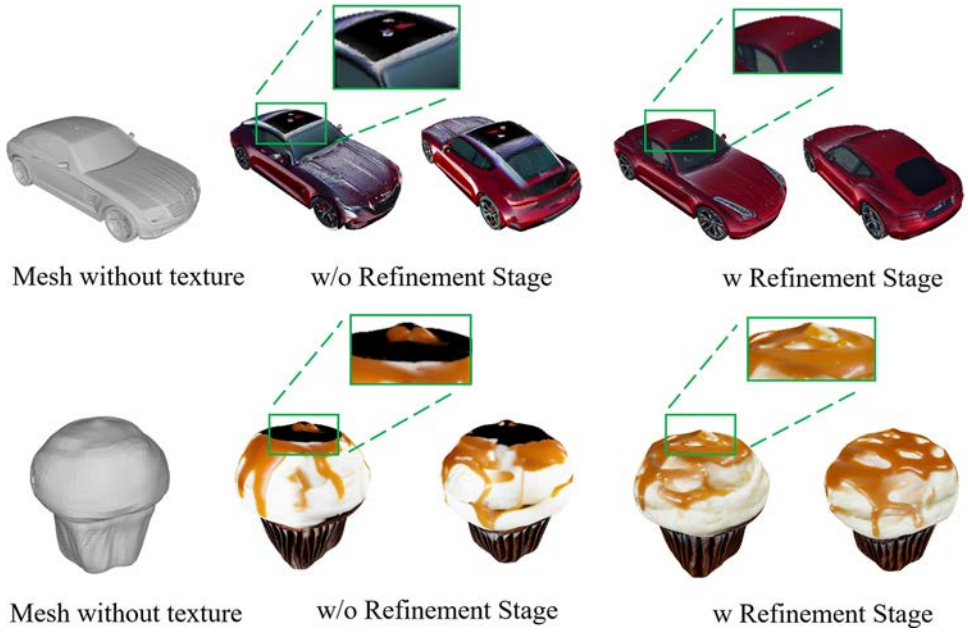


Fig. 18 Ablation study on the refinement phase

Fig. 19 Ablation study on the inclusion of text prompts

integrating text prompts into our texture generation pipeline, resulting in higher quality and more contextually appropriate textures.

5.6.5 The Coarse Phase

In Fig. 20, we illustrate the qualitative impact of employing our coarse phase versus its absence. The ablation experiments clearly demonstrate the significant impact of the coarse phase on achieving multi-view consistency in the generated textures. For example, in the case of the truck, the absence of the coarse phase results in noticeable inconsistencies between the textures on the left and right sides of the vehicle. Similarly, the texture continuity of the dumpster is disrupted without the coarse phase, displaying incongruent materials on its front and back. In contrast, when the coarse phase is incorporated, these issues are effectively resolved, leading to high texture consistency across different views. These findings validate the critical role of the coarse phase in enhancing multi-view consistency, thereby substantiating its necessity in our texture generation framework.

5.6.6 The Refinement Phase

In Fig. 18, we illustrate the qualitative impact of employing our refinement phase versus its absence. As can be seen, while the textures generated in the initial phase exhibit multi-view consistency, issues persist, such as the inability to cover the top area, leaving it black. Additionally, the texture projection back to the original image is not optimal. For instance, using

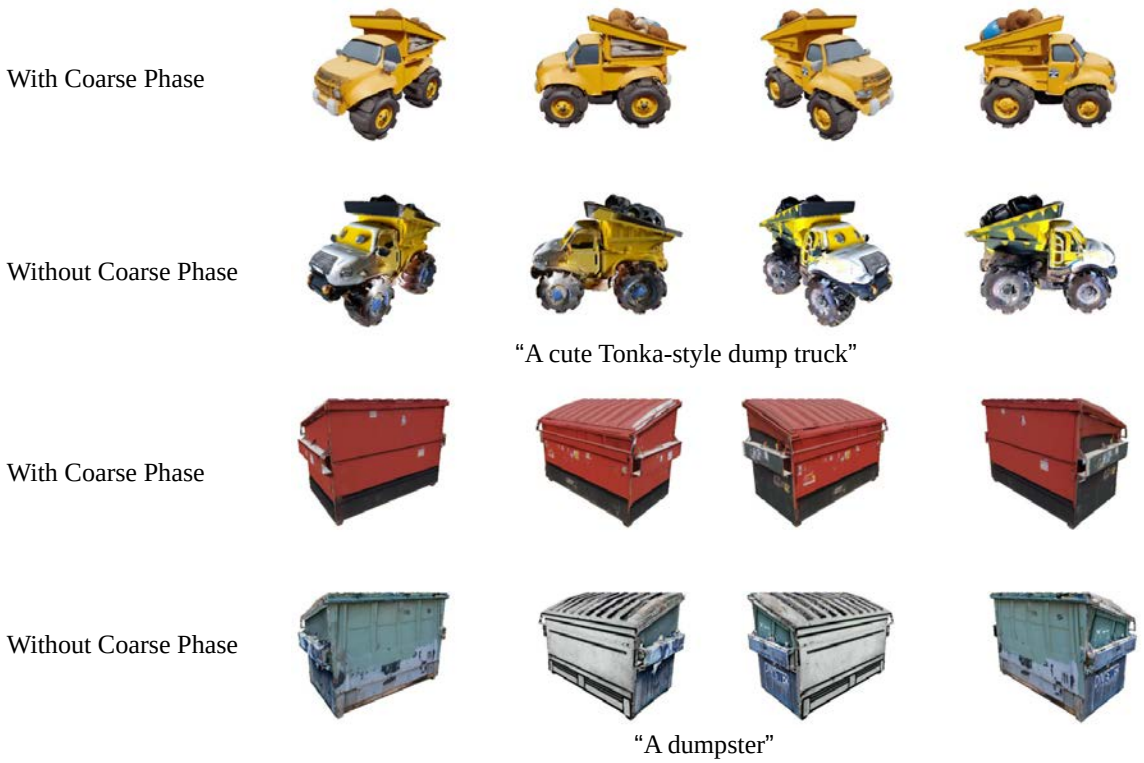


Fig. 20 Ablation study on the coarse phase in texture generation: The figure compares textures generated with and without the coarse phase. Omitting the coarse phase results in noticeable inconsistencies across views, as seen in the truck and dumpster examples. Including it enhances multi-view texture coherence, demonstrating its critical role in our model

Table 12 Ablation study on coarse phase and refinement phase

Coarse phase	Refinement phase	FID↓	KID↓
✓	✓	35.36	4.54
✓	×	47.58	14.36
×	✓	41.15	10.64

ordinary images for specific parts like the car's hood does not yield the best results. This comparison in Fig. 18 underscores the significance of refinement phase in addressing these issues to acquire a cohesive texture map. A quantitative evaluation in Tab. 12 further verifies the effectiveness of Multi-View Diffusion priors and the refinement phase.

6 Limitations and Conclusions

In this work, our study addresses the challenges faced by generative models in textured mesh generation by introducing a unified framework that combines mesh generation and texture generation. We introduced a novel sparse latent point diffusion model that enhances geometric control and resolves multi-view texture inconsistencies. By employing point clouds as an intermediate representation and integrating multi-view priors for texture generation, we achieved significant improvements in geometric quality, control, and the generation of coherent, high-quality textures. Our framework's distinct separation of control mechanisms for geometry and texture ensures visually appealing and detailed 3D content creation. Evaluations on ShapeNet and Objaverse datasets validate the superior performance of our model compared to existing approaches, highlighting its potential for advancing computer graphics and virtual content creation.

However, our algorithm does face certain limitations, particularly in terms of geometric and textural failures:

Geometric Failures: These are largely attributed to the surface reconstruction step of the process. The primary issues encountered include the occurrence of local holes and overly smooth surfaces. These deficiencies stem from the inherent limitations of point cloud-based surface reconstruction methods, which struggle with accurately capturing complex geometries or maintaining detailed surface textures under certain conditions. Figure 21 illustrates cases in which geometric anomalies appear, affecting the fidelity and integrity of the reconstructed models.

Textural Failures: Our textural generation is limited by resolution constraints, as our multi-view diffusion approach,

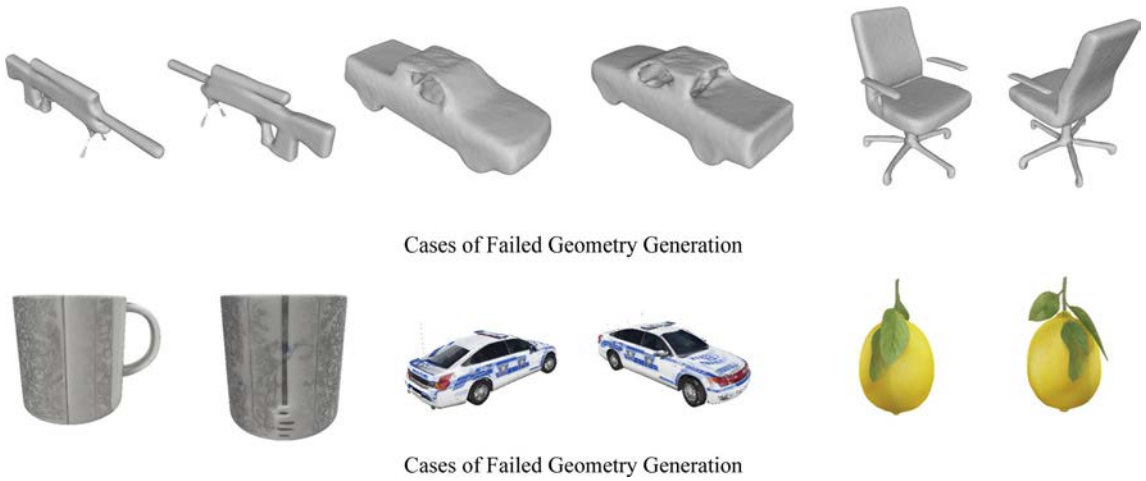


Fig. 21 Some examples of failures in geometric and texture generation

while effective for consistency across angles, cannot produce high-resolution images. This leads to less detailed textures, especially noticeable under close inspection or in high-resolution uses. Figure 21 displays examples where these limitations are apparent, highlighting the difficulties in preserving textural fidelity. Additionally, our method's weighting of different viewpoints varies with the dot product of the viewpoint direction and the surface normal, which causes blurring at seams, as shown in the Fig. 21.

Recognizing these challenges, we are actively refining our approach for future versions. We intend to transition from traditional mesh generation techniques to more advanced technologies like DMTet, which are better equipped to handle complex structures. Additionally, to enhance the scalability and effectiveness of our model, we plan to incorporate both 3D and 2D data during the training phase. These strategic improvements are expected to substantially enhance the robustness and practicality of our method, addressing the current limitations and setting a new standard for 3D content creation.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11263-024-02326-x.

Acknowledgements This work is supported by the National Key R&D Program of China (No. 2022ZD0160702), STCSM (Nos. 22511106101, 22DZ2229005), 111 plan (No. BP0719010) and National Natural Science Foundation of China (No. 62306178).

References

Anciukevicius, T., Manhardt, F., Tombari, F., & Henderson, P. (2024). Denoising diffusion via image-based rendering. Preprint retrieved from arXiv:2402.03445

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

Bokhovkin, A., Tulsiani, S., & Dai, A. (2023) Mesh2tex: Generating mesh textures from image queries. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 8918–8928)

Cai, R., Yang, G., Averbuch-Elor, H., Hao, Z., Belongie, S., Snavely, N., & Hariharan, B. (2020). Learning gradient fields for shape generation. In A. Vedaldi, H. Bischof, T. Brox et al. (Eds.) *Computer Vision-ECCV 2020-16th European conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III, Lecture Notes in Computer Science* (Vol. 12348, pp. 364–381). Springer. https://doi.org/10.1007/978-3-030-58580-8_22

Cao, T., Kreis, K., Fidler, S., Sharp, N., & Yin, K. (2023). Texfusion: Synthesizing 3d textures with text-guided image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 4169–4181)

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., & Karras, T. (2022). Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 16123–16133).

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., & Xiao, J. (2015). Shapenet: An information-rich 3d model repository. Preprint retrieved from arXiv:1512.03012

Chen, D. Z., Siddiqui, Y., Lee, H. Y., Tulyakov, S., & Nießner, M. (2023a). Text2tex: Text-driven texture synthesis via diffusion models. Preprint retrieved from arXiv:2303.11396

Chen, Q., Chen, Z., Zhou, H., & Zhang, H. (2023b) ShaDDR: Real-time example-based geometry and texture generation via 3d shape detailization and differentiable rendering. Preprint retrieved from arXiv:2306.04889

Chen, R., Chen, Y., Jiao, N., & Jia, K. (2023c) Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. Preprint retrieved from arXiv:2303.13873

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

Chen, Y., Zhang, C., Yang, X., Cai, Z., Yu, G., Yang, L., & Lin, G. (2023d). It3d: Improved text-to-3d generation with explicit view synthesis. Preprint retrieved from arXiv:2308.11473

Chou, G., Bahat, Y., & Heide, F. (2022). Diffusionsdf: Conditional generative modeling of signed distance functions. Preprint retrieved from arXiv:2211.13757. https://api.semanticscholar.org/CorpusID:254017862

Collins, J., Goel, S., Deng, K., Luthra, A., Xu, L., Gundogdu, E., Zhang, X., Vicente, T. F. Y., Dideriksen, T., Arora, H., & Guillaumin, M. (2022). Abo: Dataset and benchmarks for real-world 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 21126–21136).

Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., Farhadi, A. (2022). Objaverse: A universe of annotated 3d objects. Preprint retrieved from arXiv:2212.08051

Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., & Farhadi, A. (2023). Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 13142–13153).

Gao, J., Shen, T., Wang, Z., Chen, W., Yin, K., Li, D., Litany, O., Gojcic, Z., & Fidler, S. (2022). Get3d: A generative model of high quality 3d textured shapes learned from images. *Advances in Neural Information Processing Systems*, *35*, 31841–31854.

Gupta, A., Xiong, W., Nie, Y., Jones, I., & Oğuz, B. (2023). 3dgen: Triplane latent diffusion for textured mesh generation. Preprint retrieved from arXiv:2303.05371

Henderson, P., Tsiminaki, V., & Lampert, C. H. (2020). Leveraging 2d data to learn textured 3d mesh generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7498–7507).

Ho, J., Jain, A., & Abbeel, P. (2020a). Denoising diffusion probabilistic models. Preprint retrieved from arXiv:2006.11239

Ho, J., Jain, A., & Abbeel, P (2020b) Denoising diffusion probabilistic models. In: H. Larochelle, M. Ranzato, R. Hadsell, et al. (Eds.) *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020, NeurIPS 2020, December 6–12, 2020, virtual*. https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html

Höllein, L., Božič, A., Müller, N., Novotny, D., Tseng, H.Y., Richardt, C., Zollhöfer, M., & Nießner, M. (2024). Viewdiff: 3d-consistent image generation with text-to-image models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5043–5052).

Hong, F., Zhang, M., Pan, L., Cai, Z., Yang, L., & Liu, Z. (2022). Avatarclip: Zero-shot text-driven generation and animation of 3d avatars. Preprint retrieved from arXiv:2205.08535

Huang, J., Thies, J., Dai, A., Kundu, A., Jiang, C., Guibas, L.J., Nießner, M., & Funkhouser, T. (2020). Adversarial texture optimization from rgb-d scans. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 1559–1568).

Jun, H., & Nichol, A. (2023). Shap-e: Generating conditional 3d implicit functions. Preprint retrieved from arXiv:2305.02463

Karnewar, A., Mitra, N. J., Vedaldi, A., & Novotny, D. (2023). Holofusion: Towards photo-realistic 3d generative modeling. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 22976–22985).

Kopf, J., Fu, C. W., Cohen-Or, D., Deussen, O., Lischinski, D., & Wong, T. T. (2007). Solid texture synthesis from 2d exemplars. In *ACM SIGGRAPH 2007 papers* (p 2–es).

Lefebvre, S., & Hoppe, H. (2006). Appearance-space texture synthesis. *ACM Transactions on Graphics (TOG)*, *25*(3), 541–548.

Li, M., Duan, Y., Zhou, J., & Lu, J. (2022a). Diffusion-sdf: Text-to-shape via voxelized diffusion. In *2023 IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 12642–12651). https://api.semanticscholar.org/CorpusID:254366593

Li, Y., Dou, Y., Shi, Y., Lei, Y., Chen, X., Zhang, Y., Zhou, P., & Ni, B. (2023). Focaldreamer: Text-driven 3d editing via focal-fusion assembly. Preprint retrieved from arXiv:2308.10608

Li, Y., Upadhyay, U., Slim, H., Abdelreheem, A., Prajapati, A., Pothigara, S., Wonka, P., & Elhoseiny, M. (2022b). 3d compat: Composition of materials on parts of 3d things. In *European conference on computer vision* (pp. 110–127). Springer.

Lin, C. H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M. Y., Lin, T. Y. (2023). Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 300–309).

Liu, Y., Xie, M., Liu, H., & Wong, T. T.(2023a). Text-guided texturing by synchronized multi-view diffusion. Preprint retrieved from arXiv:2311.12891

Liu, Z., Feng, Y., Black, M. J., Nowrouzezahrai, D., Paull, L., & Liu, W. (2023b). Meshdiffusion: Score-based generative 3d mesh modeling. Preprint retrieved from arXiv:2303.08133, https://api.semanticscholar.org/CorpusID:257505014

Lorensen, W. E., & Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. https://api.semanticscholar.org/CorpusID:15545924

Luo, S., & Hu, W. (2021a). Diffusion probabilistic models for 3d point cloud generation. In *2021 IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 2836–2844). https://api.semanticscholar.org/CorpusID:232092778

Luo, S., & Hu, W. (2021b). Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2837–2845).

Luo, T., Rockwell, C., Lee, H., & Johnson, J. (2023). Scalable 3d captioning with pretrained models. Preprint retrieved from arXiv:2306.07279

Lyu, Z., Kong, Z., Xu, X., Pan, L., & Lin, D. (2021). A conditional point diffusion-refinement paradigm for 3d point cloud completion. Preprint retrieved from arXiv:2112.03530

Lyu, Z., Wang, J., An, Y., Zhang, Y., Lin, D., & Dai, B. (2023). Controllable mesh generation through sparse latent point diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 271–280).

Ma, Y., Zhang, X., Sun, X., Ji, J., Wang, H., Jiang, G., Zhuang, W., & Ji, R. (2023). X-mesh: Towards fast and accurate text-driven 3d stylization via dynamic textual guidance. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 2749–2760).

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., & Geiger, A. (2018). Occupancy networks: Learning 3d reconstruction in function space. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 4455–4465). https://api.semanticscholar.org/CorpusID:54465161

Metzer, G., Richardson, E., Patashnik, O., Giryes, R., & Cohen-Or, D. (2023). Latent-nerf for shape-guided generation of 3d shapes and textures. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 12663–12673).

Michel, O., Bar-On, R., Liu, R., Benaim, S., & Hanocka, R. (2022). Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 13492–13502).

Mohammad Khalid, N., Xie, T., Belilovsky, E., & Popa, T. (2022). Clip-mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia 2022 conference papers* (pp. 1–8).

Nam, G., Khlifi, M., Rodriguez, A., Tono, A., Zhou, L., & Guerrero, P. (2022). 3d-ldm: Neural implicit 3d shape generation with latent diffusion models. Preprint retrieved from arXiv:2212.00842, https://api.semanticscholar.org/CorpusID:254220714

Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., & Chen, M. (2022a). Point-e: A system for generating 3d point clouds from complex prompts. Preprint retrieved from arXiv:2212.08751, https://api.semanticscholar.org/CorpusID:254854214

Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., & Chen, M. (2022b). Point-e: A system for generating 3d point clouds from complex prompts. Preprint retrieved from arXiv:2212.08751

Oechsle, M., Mescheder, L., Niemeyer, M., Strauss, T., & Geiger, A. (2019). Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 4531–4540).

Pan, Z., Lu, J., Zhu, X., & Zhang, L. (2023). Enhancing high-resolution 3d generation through pixel-wise gradient clipping. Preprint retrieved from arXiv:2310.12474

Park, J. J., Florence, P., Straub, J., Newcombe, R., & Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 165–174). https://api.semanticscholar.org/CorpusID:58007025

Peng, S., Jiang, C., Liao, Y., Niemeyer, M., Pollefeys, M., & Geiger, A. (2021). Shape as points: A differentiable Poisson solver. *Advances in Neural Information Processing Systems*, *34*, 13032–13044.

Peng, S., Jiang, C., Liao, Y., Niemeyer, M., Pollefeys, M., & Geiger, A. (2021b). Shape as points: A differentiable Poisson solver. In *Neural information processing systems*. https://api.semanticscholar.org/CorpusID:235358422

Poole, B., Jain, A., Barron, JT., & Mildenhall, B. (2022). Dreamfusion: Text-to-3d using 2d diffusion. Preprint retrieved from arXiv:2209.14988

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: I. Guyon, U. V. Luxburg, S. Bengio, et al. (Eds.) *Advances in Neural Information Processing Systems* (Vol. 30). Curran Associates, Inc., https://proceedings.neurips.cc/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf

Qian, G., Mai, J., Hamdi, A., Ren, J., Siarohin, A., Li, B., Lee, H.Y., Skorokhodov, I., Wonka, P., Tulyakov, S., & Ghanem, B. (2023). Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. Preprint retrieved from arXiv:2306.17843

Raj, A., Kaza, S., Poole, B., Niemeyer, M., Ruiz, N., Mildenhall, B., Zada, S., Aberman, K., Rubinstein, M., Barron, J., & Li, Y. (2023). Dreambooth3d: Subject-driven text-to-3d generation. Preprint retrieved from arXiv:2303.13508

Rakotosaona, M.J., Manhardt, F., Arroyo, D.M., Niemeyer, M., Kundu, A., & Tombari, F. (2024). Nerfmeshing: Distilling neural radiance fields into geometrically-accurate 3d meshes. In *2024 international conference on 3D vision (3DV)* (pp. 1156–1165). IEEE.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. Preprint retrieved from arXiv:2204.06125 *1*(2), 3.

Richardson, E., Metzer, G., Alaluf, Y., Giryes, R., & Cohen-Or, D. (2023). Texture: Text-guided texturing of 3d shapes. Preprint retrieved from arXiv:2302.01721

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10684–10695).

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., & Ho, J. (2022). Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, *35*, 36479–36494.

Shen, T., Gao, J., Yin, K., Liu, M. Y., & Fidler, S. (2021). Deep marching tetrahedra: A hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, *34*, 6087–6101.

Shi, Y., Wang, P., Ye, J., Long, M., Li, K., & Yang, X. (2023). Mvdream: Multi-view diffusion for 3d generation. Preprint retrieved from arXiv:2308.16512

Shue, J.R., Chan, E.R., Po, R., Ankner, Z., Wu, J., & Wetzstein, G. (2022). 3d neural field generation using triplane diffusion. In *2023 IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 20875–20886). https://api.semanticscholar.org/CorpusID:254095843

Siddiqui, Y., Thies, J., Ma, F., Shan, Q., Nießner, M., & Dai, A. (2022). Texturify: Generating textures on 3d shape surfaces. In *European Conference on Computer Vision* (pp. 72–88). Springer.

Song, J., Meng, C., & Ermon, S. (2020) Denoising diffusion implicit models. In *International Conference on Learning Representations*.

Szymanowicz, S., Rupprecht, C., & Vedaldi, A. (2023) Viewset diffusion:(0-) image-conditioned 3d generative models from 2d data. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 8863–8873).

Tang, J., Wang, T., Zhang, B., Zhang, T., Yi, R., Ma, L., & Chen, D. (2023). Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior. Preprint retrieved from arXiv:2303.14184

Turk, G. (2001). Texture synthesis on surfaces. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (pp. 347–354).

Wei, L. Y., Lefebvre, S., Kwatra, V., & Turk, G. (2009). State of the art in example-based texture synthesis. Eurographics 2009, State of the Art Report, EG-STAR (pp. 93–117).

Wei, L. Y., & Levoy, M. (2001). Texture synthesis over arbitrary manifold surfaces. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (pp. 355–360).

Yang, B., Dong, W., Ma, L., Hu, W., Liu, X., Cui, Z., & Ma, Y. (2023). Dreamspace: Dreaming your room space with text-driven panoramic texture propagation. Preprint retrieved from arXiv:2310.13119

Yang, G., Huang, X., Hao, Z., Liu, M.Y., Belongie, S., & Hariharan, B. (2019). Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 4541–4550).

Zeng, X., Vahdat, A., Williams, F., Gojcic, Z., Litany, O., Fidler, S., & Kreis, K. (2022a). Lion: Latent point diffusion models for 3d shape generation. Preprint retrieved from arXiv:2210.06978. https://api.semanticscholar.org/CorpusID:252872881

Zeng, X., Vahdat, A., Williams, F., Gojcic, Z., Litany, O., Fidler, S., & Kreis, K. (2022b). Lion: Latent point diffusion models for 3d shape generation. Preprint retrieved from arXiv:2210.06978

Zheng, X., Liu, Y., Wang, P., & Tong, X. (2022). Sdf-stylegan: Implicit sdf-based stylegan for 3d shape generation. In *Computer Graphics Forum* (pp. 52–63). Wiley Online Library.

Zheng, X. Y., Pan, H., Wang, P. S., Tong, X., Liu, Y., & Shum, H. Y. (2023). Locally attentional sdf diffusion for controllable 3d shape generation. *ACM Transactions on Graphics (TOG), 42*, 1–13.

Zhou, L., Du, Y., & Wu, J. (2021a) 3d shape generation and completion through point-voxel diffusion. In *2021 IEEE/CVF international conference on computer vision (ICCV)* (pp. 5806–5815). https://api.semanticscholar.org/CorpusID:233182041

Zhou, L., Du, Y., & Wu, J. (2021b) 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 5826–5835).

Zhou, Q. Y., & Koltun, V. (2014). Color map optimization for 3d reconstruction with consumer depth cameras. *ACM Transactions on Graphics (ToG), 33*(4), 1–10.

Zhuang, J., Wang, C., Lin, L., Liu, L., & Li, G. (2023). Dreameditor: Text-driven 3d scene editing with neural fields. Preprint retrieved from arXiv:2306.13455

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.