ShapeShifter: 3D Variations Using Multiscale and Sparse Point-Voxel Diffusion

Nissim Maruani
Inria, Université Côte d'Azur
nissim.maruani@inria.fr

Wang Yifan
Adobe Research
yifwang@adobe.com

Matthew Fisher
Adobe Research
matfishe@adobe.com

Pierre Alliez
Inria, Université Côte d'Azur
pierre.alliez@inria.fr

Mathieu Desbrun
Inria Saclay - Ecole Polytechnique
mathieu.desbrun@inria.fr

Abstract

This paper proposes ShapeShifter, a new 3D generative model that learns to synthesize shape variations based on a single reference model. While generative methods for 3D objects have recently attracted much attention, current techniques often lack geometric details and/or require long training times and large resources. Our approach remedies these issues by combining sparse voxel grids and point, normal, and color sampling within a multiscale neural architecture that can be trained efficiently and in parallel. We show that our resulting variations better capture the fine details of their original input and can handle more general types of surfaces than previous SDF-based methods. Moreover, we offer interactive generation of 3D shape variants, allowing more human control in the design loop if needed.

1. Introduction

Creating 3D content through generative models is currently attracting significant attention. Traditional 3D modeling demands both time and specialized skills to create complex shapes, whereas advancements in generative AI promise a broader exploration of design possibilities, free from the usual constraints of time or technical expertise. However, current 3D generative models have numerous shortcomings that limit their utility in applications such as movies, gaming, and product design. First, state-of-the-art methods often struggle to produce the fine geometric details and sharp features necessary for digital shapes to be practical in geometric modeling. Additionally, these models require large, high-quality 3D datasets, which are significantly more challenging to curate compared to image datasets, and involve long training times and substantial computational resources.

In this paper, we tackle a task-specific, yet effective approach to synthesizing geometry: we propose generating shape variations from a single high-quality example. This lesser-explored generative method offers several benefits beyond avoiding the curation of large training datasets: it has the potential to provide a resource-efficient way to generate shape variants for retargeting or editing, automatically inheriting the style, symmetries, semantics, and geometric details from the exemplar. Although existing generative methods from exemplars are able to create varied 3D assets, they struggle to produce clean and detailed geometry due to their reliance on occupancy fields [50, 92, 106] or signed distance functions [107] (which smooth out geometric features), or because they are supervised through volumetric

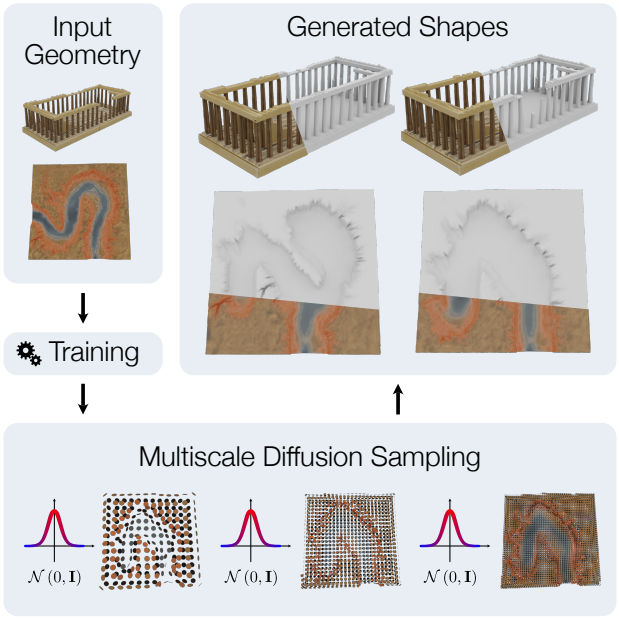
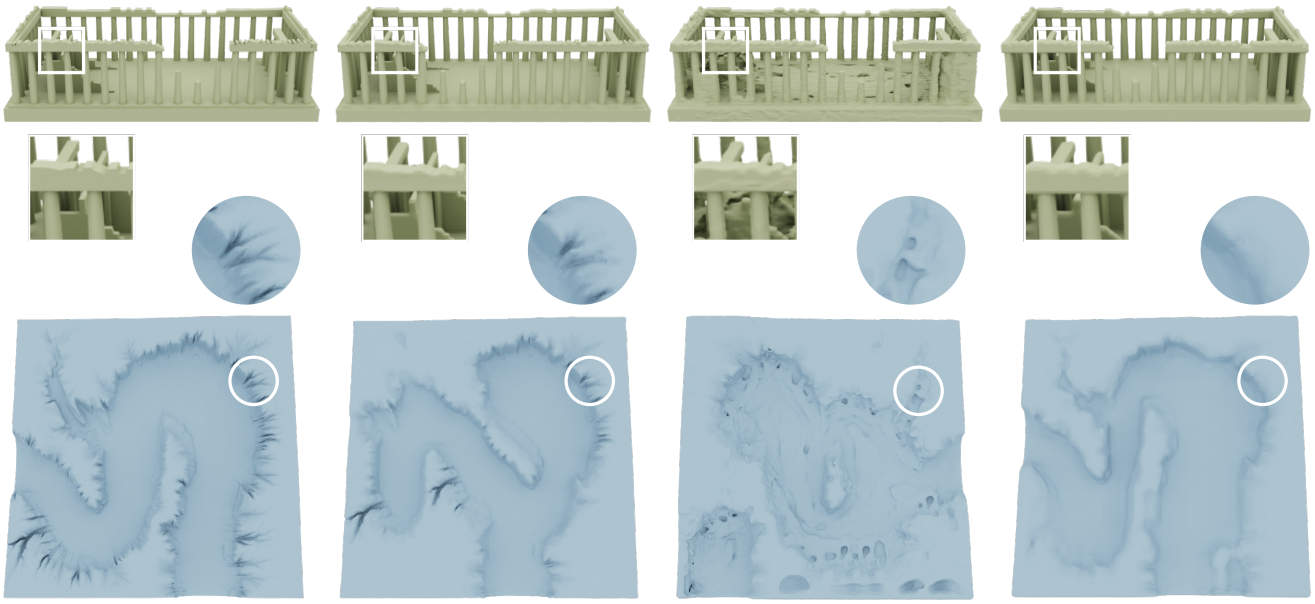


Figure 1. We propose ShapeShifter. Given a 3D exemplar, we train a hierarchical diffusion model to create novel variations that preserve the geometric details and styles of the exemplar. By combining compact yet explicit 3D features (as colored, oriented points) with a sparse voxel grid, we shorten training times from hours to minutes, while yielding significantly better geometric quality than prior work. The hierarchical point representation and fast inference times further enables intuitive interactive editing.



(a) Input Geometry (b) Ours (c) Sin3DGen (d) Sin3DM

Figure 2. Geometric details. Our generation captures significantly more geometric details present in the exemplar mesh (leftmost). Prior work, Sin3DGen [50] and Sin3DM [107], operates with plenoxels and neural radiance fields encoded in single-resolution triplane features, respectively, which lack the capability to sufficiently represent and supervise high-resolution geometric details. In contrast, our method employs a colored and oriented point set, providing precise geometric information .

rendering [50, 92] (which often leads to large geometric artifacts) — and without a clean geometric output model, the use of 2D textures to further enhance visual complexity is severely hindered. Consequently, existing exemplar-based methods are relative slow in generating variations of the input as they must rely on volumetric sampling within the surface's neighborhood [92, 106, 107].

We propose a novel approach, which we call ShapeShifter, to synthesize high-quality shape variations of an input 3D model, with training and inference times well suited for practical real-world applications. We use points (with their normals and optionally colors for additional semantic information) as our lightweight and efficient base geometric representation [77], which we pair with a multiscale 3D diffusion network. While these explicit surface features already streamline the generative process and help preserve geometric details, we propose to significantly reduce training times and achieve interactive inference rates by adopting sparse convolutions based on fVDB [105], a recent spatial learning framework based on sparse voxel grids. Mixing point sampling and sparse convolution, a novel combination in generative modeling, results in a multiscale generative approach capable of producing 3D variants of shapes of different styles and topologies. Furthermore its fast inference allows for interactive editing control.

Contributions. This paper proposes a neural network approach to generating high-quality shapes from a single 3D reference example. Compared to previous exemplar-based generative methods, we demonstrate significantly improved geometric quality of our outputs, as shown in Fig. 2. Moreover, the simplicity of our geometric representation (using point sampling in a sparse voxel grid) and its hierarchical refinement (learned per level in parallel) to control and generate variations of an arbitrary closed or open input shape results in significantly reduced training times (minutes instead of hours) and interactive inference. While our results can be easily converted into textured meshes, direct visualization of our point-based representation in realtime enables iterative co-creation guided by an artist. Finally, our high-quality output geometric models can be assigned a fine texture if needed, using off-the-shelf image super-resolution or more advanced texturing synthesis methods such as [84].

2. Related Work

3D Generation. The field of 3D generation has seen rapid development in recent years. Advances in generative models and large-scale 3D datasets have underpinned this progress. Generative adversarial networks (GANs) [23] have been widely used in works like [2, 10, 22], while normalizing flows [83] were utilized in [111]. Other approaches include variational autoencoders (VAEs) [45] and autoregressive (AR) models [6, 25, 97], as shown in [14, 70, 74, 90, 114, 116]. The recent introduction of diffusion models [33, 91] has enabled training on larger datasets such as Objaverse [18]. A survey by Po et al. [3] provides a comprehensive taxonomy of 3D diffusion approaches. A primary line of work builds on 2D diffusion models, generat-

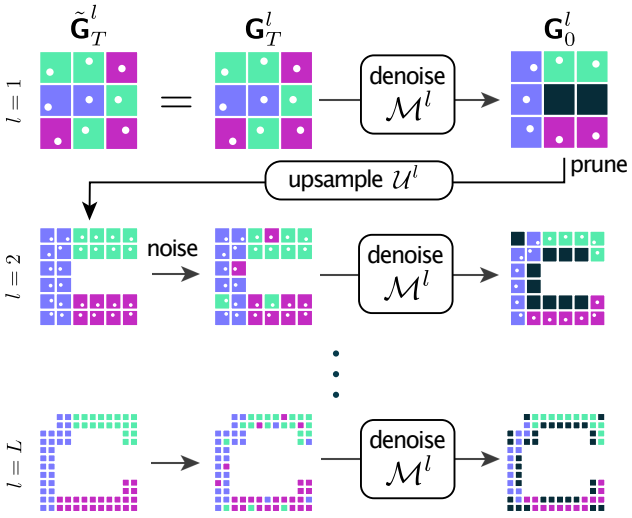


Figure 3. Multiscale diffusion on sparse voxel grid. We start from noise $\epsilon \sim \mathcal{C}(\mathbf{0}, \boldsymbol{I})$ at the coarsest level $l=1$, and obtain the 3D feature grid $\mathbf{G}^l$ through reverse diffusion. Each subsequent level uses the output of the previous level. Inactive voxels are first pruned, then upsampled with a level-specific upsampler $\mathcal{U}^l$. The upsampled grid $\tilde{\mathbf{G}}^l$ is subsequently noised and passed through the diffusion model to obtain a clean version of the sparse feature grid $\mathbf{G}^l$. All levels are independent and can thus be trained in parallel.

ing multiview-consistent images through Score Distillation Sampling (SDS) [76, 98]. However, SDS faces practical challenges such as high optimization times [52, 67], color artifacts [12, 51, 61, 102], and 3D inconsistencies [57, 101]. Fine-tuning diffusion models on 3D assets for direct multiview output [53, 55, 59, 79, 88] can address these issues, with further speedups through reconstructor networks for radiance fields [11, 49, 54, 100, 103] or Gaussian splats [109, 118, 123]. However, photometric losses often lead to geometric artifacts. A separate direction directly trains 3D diffusion models on 3D data [62, 71] or encodes 3D data through autoencoders [17, 27, 39, 81, 82, 96, 117, 119, 120]. These methods demand extensive, high-quality data and substantial computational resources, and the generated geometry, while improved, still lacks the geometric details required in real-world 3D applications. We adopt an alternative approach, generating 3D assets with high-quality geometry from a single exemplar, trainable on a single GPU in minutes, while enabling user control over output shapes.

Generation from a single instance. Despite advancements in large-scale 3D generation, high-quality 3D data remains scarce, and computational costs for training and inference are significant. Generating 3D content from a single high-quality example offers an appealing alternative, giving users control through concrete exemplar inputs. Non-learning, patch-based methods such as PatchMatch [5, 29] and more recent works [20, 24] produce variations by finding and blending similar patches within the exemplar. In contrast, SinGAN [87] and its successors [32, 89] have shown that generative models can be trained on a single example. More recently, SinFusion [46, 72] demonstrated that diffusion models, known for their stability, can also be adapted to this approach. Applications of these single-instance models include texture synthesis [69, 73, 85, 122], video synthesis [28], and more.

In 3D generation, similar approaches have emerged. Herz et al. [31] applied SinGAN with mesh convolutions [30] to enhance surface details without altering structure, while works like Son et al. [92] and Karnewar et al. [41] generated radiance fields from a single instance. Wu et al. [106] used 3D convolutions to generate significant variations, later extending their work to include texture synthesis [107], though details are often degraded due to 3D convolution bottlenecks. These methods, however, are typically slow, with training times ranging from 2 to 4 hours. To bypass training, Li et al. [50] adopted a PatchMatch-based method for 3D scenes represented by plenoxels [21]. Although this removes the training step, it still requires approximately 10 minutes per variation and struggles to maintain geometric quality due to challenges in converting the plenoxels' occupancy fields to signed distance functions. By combining the strengths of a recent framework for learning sparse spatial information [105] with compact, geometry-centric features, our method achieves higher geometric fidelity while requiring less than 12 minutes for the entire generation process (including training and inference).

3D Representation. Earlier 3D representations focused solely on geometry, using formats like voxels [108], 3D points [78], meshes [30, 64, 65], SDFs [74], and occupancy fields [15, 66]. Differentiable rendering later enabled joint modeling of geometry and appearance, spanning from neural mesh rasterization [42, 56] and splatting methods [44, 104, 113] to neural radiance fields [58, 68]. This integration allowed the use of 2D assets [10, 86] and facilitated large-scale generation and reconstruction [34, 94, 103, 109, 110], enhancing semantic context for geometry generation. However, appearance signals often exhibit higher frequencies than geometry, leading existing methods to separate texture and geometry branches [68]. Tying texture generation to geometry imposes unnecessarily high resolution requirement on geometry, slowing single-exemplar generation [107]. Other works generate geometry first, before refining the appearance in 2D texture space at higher resolutions [36, 112, 115]. Recently, Clay [119] fully separates geometry and appearance aspects, achieving state-of-the-art quality in both. Following this idea, we focus on high-fidelity geometry generation, which is the weakest aspect of current 3D generation approaches. Unlike Clay's strict separation, however, we use RGB features to add contextual information-—crucial for single-exemplar generation where data priors are weaker.

Our results show that this approach produces superior geometry which, when combined with state-of-the-art image super-resolution [40, 48] and texture synthesis [8, 84], allows for highly detailed shapes with sharp geometry and HD textures. We argue that this design, for a fixed computational budget, optimally balances quality and efficiency.

3. Method

Our goal is to train a generative model from a single 3D input mesh to generate new variations efficiently. We use a multiscale diffusion model with limited receptive fields to learn the internal structures of the given shape, adapting an approach that has been used for training a generative model on a single image [46]. We use compact, explicit 3D features directly extracted from the exemplar shape for diffusion. These features are encoded in a sparse voxel grid, and processed efficiently using a specialized 3D convolution framework (fVDB [105]) to capture fine-scale geometric details without incurring high memory cost. We introduce our 3D features in Sec. 3.1, the hierarchical diffusion model in Sec. 3.2, and the final meshing process of a generated output in Sec. 3.3.

3.1. 3D Representation

Explicit Multiscale 3D Features. Our method employs explicit 3D information to encode the geometry of the input exemplar mesh at multiple scales. They are composed of merely 10 values per voxel of a sparse voxel grid,

$$\boldsymbol{f} = (\boldsymbol{p}_x, \boldsymbol{p}_y, \boldsymbol{p}_z, \boldsymbol{n}_x, \boldsymbol{n}_y, \boldsymbol{n}_z, \boldsymbol{c}_r, \boldsymbol{c}_g, \boldsymbol{c}_b, m), \qquad (1)$$

where $(\boldsymbol{p}_x, \boldsymbol{p}_y, \boldsymbol{p}_z) \in [-0.5, 0.5]^3$ represents a point sample encoded as an offset from the voxel center, $(\boldsymbol{n}_x, \boldsymbol{n}_y, \boldsymbol{n}_z) \in [-1, 1]^3$ represents the local normal of the underlying geometry associated to the point sample, $(\boldsymbol{c}_r, \boldsymbol{c}_g, \boldsymbol{c}_b) \in [-2, 2]^3$ represents the color scaled up to the $[-2, 2]$ range, and $m \in [-1, 1]$ is a scalar indicating if the voxel contains the mesh surface which we will use as a mask to prune voxels after refinement (see Sec. 3.2). The value ranges of the feature components were empirically chosen since feature scale can be important in diffusion models [13].

These features are extracted from the different scales of the input mesh. Specifically, starting from the finest scale L, for each voxel that intersects the surface, we find the nearest surface point to the voxel center, whose position, normal, and color are used to form the feature. The mask is set to 1 for all selected voxels as they contain the surface. At each subsequent scale $l < L$, we obtain coarser points, normals, and colors features through a 2^3 average pooling, and the mask value through max pooling. To better preserve sharp features, we average the point positions in the Quadric Error Metric sense [65] (see supplemental material for details).

This 3D representation yields three advantages:

Algorithm 1: Training at level l

Input: Extracted sparse features $\{\mathbf{G}^1, \cdots, \mathbf{G}^L\}$
// Upsampler Training for Levels 2 to L
repeat
  Update model $\mathcal{U}^l$ with the loss Eq. (4)
until convergence;
// Diffusion Model Training
repeat
  $t \sim \text{Uniform}(0, T)$
  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
  if $l = 1$ **then**
    $\mathbf{G}^{l,\text{mix}} = \mathbf{G}^1$
  else
    $\tilde{\mathbf{G}}^l = \mathcal{U}^l\left(\mathbf{G}^{l-1}\right)$
    $\mathbf{G}^{l,\text{mix}} = \gamma(t)\mathbf{G}^l + (1-\gamma(t))\tilde{\mathbf{G}}^l$
  $\mathbf{G}^l_t = \sqrt{\bar{\alpha}(t)}\mathbf{G}^{l,\text{mix}} + \sqrt{1-\bar{\alpha}(t)}\epsilon$
  Update model with $\|\mathcal{M}^l(\mathbf{G}^l_t|t) - \mathbf{G}^l\|^2$.
until convergence;

- it captures surface details in a compact form and carries contextual information from the texture;
- it encodes the 3D shape explicitly at each level, which enables a generated shape to be easily visualized or even edited at every level in a coarse-to-fine fashion;
- it is extracted from an input exemplar efficiently and deterministically, and will allow us to train each level of our model in parallel.

Sparse Voxel Grid. The inductive biases of convolutional neural networks exploit the shared information across internal crops within the input data, which is essential to learning from a single example and prevents overfitting [46, 72, 87]. However, 3D convolutional operations are notoriously expensive computationally and memory intensive. To address this issue, we leverage fVDB [105], a recently proposed framework that supports efficient operations on sparse voxels. As a result, only active features are stored and processed, which significantly reduces the memory footprint and computational complexity. We denote the sparse feature grid storing active 3D features at level l as $\mathbf{G}^l = \{\boldsymbol{f}^l\}$.

3.2. Multiscale Diffusion

Our multiscale diffusion pipeline generalizes SinDDM [46] to 3D and adapts it to properly work with sparse voxel grid. As shown in Fig. 3, the pipeline consists of multiple diffusion models $\{\mathcal{M}^l\}_{1 \le l \le L}$. During training, these diffusion models can be trained in parallel; at inference time, new variations are generated by sequentially running the reverse diffusion in a coarse-to-fine manner. Below, we explain the hierarchical multi-scale diffusion and highlight our design

differences compared to SinDDM.

Algorithm 2: Sampling

Input: Choice of sampler $\mathcal{S} \in \{\text{DDPM}, \text{DDIM}\}$
Output: Generated sparse grid $\mathbf{G}_L$
// Upsampler training for levels 2 to L
$\mathbf{G}_T^1 \sim \mathcal{N}(0, \mathbf{I})$
for $l \leftarrow 1$ **to** L **do**
  if $l > 1$ **then**
    $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
    $\tilde{\mathbf{G}}^l = \mathcal{U}^l(\mathbf{G}^{l-1})$
    $\mathbf{G}^l_{T[l]} = \sqrt{\bar{\alpha}(T[l])}\tilde{\mathbf{G}}^l + \sqrt{1-\bar{\alpha}(T[l])}\,\epsilon$
  for $t \leftarrow T[l]$ **to** 1 **do**
    $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
    $\mathbf{G}^l_{t-1} = \mathcal{S}(\mathcal{M}^l, \mathbf{G}^l_t, \bar{\alpha}, \epsilon, t)$
  $\mathbf{G}^l = \textbf{Prune}(\mathbf{G}^l_0)$

Forward Multiscale Diffusion. Except at its coarsest level $\mathcal{M}^1$, our diffusion model $\mathcal{M}^{l>1}$ generates the signal of the current level based on the output of the previous $(l-1)$ level. This initial guess is obtained by upsampling the output from the previous level $\tilde{\mathbf{G}}^l = \mathcal{U}(\mathbf{G}^{l-1})$, which can be seen as a "blurred" version of $\mathbf{G}^l$. This means that, for $l>1$, the diffusion model not only needs to denoise but also "deblur" during sampling. As a result, SinDDM modifies the forward diffusion process to be

$$\mathbf{G}^l_t = \sqrt{\bar{\alpha}(t)}\left(\gamma(t)\mathbf{G}^l + (1-\gamma(t))\tilde{\mathbf{G}}^l\right) + \sqrt{1-\bar{\alpha}(t)}\,\epsilon, \quad (2)$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, while $\bar{\alpha}(t)$ and $\gamma(t)$ are monotocally decreasing functions from 1 to 0 as t grows from 0 to T. The model learns to denoise the corrupted feature $\boldsymbol{f}^l_t$ at time step t by minimizing the reconstruction loss

$$\|\mathcal{M}^l(\mathbf{G}^l_t|t) - \mathbf{G}^l\|^2. \quad (3)$$

Contrasting with SinDDM which employs a bilinear upsampler as $\mathcal{U}$, we use a level-specific upsampler $\mathcal{U}^l$ motivated by the fact our spatial features (points and normals) are extracted by projecting the voxel centers onto the mesh surface — thus potentially exhibiting abrupt local changes. This results in improved preservation of sharp geometric features as we show in Sec. 5. The upsampler $\mathcal{U}^l$ is trained to minimize the L^2-loss between upsampled and ground-truth features, i.e.,

$$\|\mathcal{U}^l(\mathbf{G}^{l-1}) - \mathbf{G}^l\|^2. \quad (4)$$

Crucially, the training of different levels can be parallelized. For each level $l > 1$, we first train the upsampler and the diffusion model as summarized in Algorithm 1. Unlike SinDDM, training needs to accommodate our use of sparse grids. When comparing the denoised sparse feature grid and the ground-truth sparse feature grid (Eq. (3)), the denoised grid can contain more active voxels (see dark voxels in Fig. 3, even though their mask could be -1 — yet fVDB operations on two sparse feature grids assume that they have the same active voxels. To solve this problem, we flood those inactive voxels in the ground-truth $\mathbf{G}^l$ with feature values of the nearby active cells using a blurring kernel. All features except the mask value are flooded in this way, whereas the mask value is set to -1. Empirically, we observe that soft blending the feature values this way (instead of hard setting the values to an arbitrary number or applying an additional mask for loss) achieves the best result.

Reverse Multiscale Diffusion. Once trained, we can apply standard DDPM [33] or DDIM [93] sampling sequentially from levels 1 to L. As outlined in Algorithm 2, we start from a noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and run the reverse sampling to obtain an initial prediction at the coarsest level. Then, for each level, we first prune the predicted inactive voxels from the previous level by removing any feature entries with mask value $m < 0$. The resulting feature grid is then upsampled with $\mathcal{U}^l$, and subsequently corrupted with noise, before being given to the diffusion model for reverse sampling. Similar to SinDDM, we only add noise up to timestep $T[l] < T$ to prevent destroying the prediction from the previous level. A schematic overview of the sampling process is illustrated in Fig. 3.

3.3. Meshing

Once a new geometric variant has been created, we can directly visualize the generated shape using the points (one per finest voxel in the fVDB data structure) along with their associated normal and color. We can also trivially generate a mesh of the geometry through Poisson reconstruction [43] (or APSS [26] if we are dealing with open surfaces). One can assign colors to the mesh nodes based on the output colors, bake texture maps (as used sporadically in figures), or further refine and stylize the texture with off-the-shelf image enhancement models (see Sec. 5.5).

4. Implementation Details

Implementation Details. We implemented our method in Python with PyTorch [75], libigl [38], and Open3D [121], and our code will be made public upon acceptance. All reported timings were obtained on a desktop with an NVIDIA GeForce RTX 3080 GPU (10 GB) to underscore the efficiency of our approach even on consumer-grade hardware.

Model Parameters. By default we use 5 levels, the lowest and highest grid resolutions being 16 and 256 respectively. The upsamplers $\mathcal{U}$ consist of 4 layers of 64 channels, containing ~55k parameters that are trained for 10k iterations with a learning rate of $5 \cdot 10^{-4}$ and a 5% dropout rate.

Metric	Method	acropolis	canyon	fighting-pillar	house	ruined-tower	small-town	stone-cliff	wood	average
G-Qual. ↓	Sin3DGen	4.83	6.16	8.45	17.95	6.98	4.02	13.02	10.32	8.97
	Sin3DM	0.92	2.23	**0.26**	2.01	0.49	**0.84**	**0.02**	0.09	0.86
	ShapeShifter	**0.01**	**0.21**	**0.26**	**0.01**	**0.11**	1.00	0.10	**0.02**	**0.21**
G-Div. ↑	Sin3DGen	**0.25**	**0.50**	**0.59**	**0.41**	**0.86**	**0.70**	**0.65**	**0.44**	**0.55**
	Sin3DM	0.12	0.17	0.15	0.01	0.21	0.60	0.32	0.10	0.21
	ShapeShifter	0.04	0.19	0.24	0.01	0.32	0.60	0.23	0.08	0.21

Table 1. Evaluating geometric quality and diversity using SSFID and pairwise IoU scores. Our model shows clear advantage in quality, and performs similar to Sin3DM in diversity. As we discussed in Sec. 5.1, both metrics have their blindspots, SSFID overlooks geometric details and pairwise IoU rewards artifacts. Finding a more holistic metric to evaluate shape variation remains an open problem.

The diffusion models have 128 feature channels and 7 layers, for a total of ~565k parameters for the coarsest model $\mathcal{M}^1$ and 1.2M parameters As in prior work, the receptive fields of each model $\mathcal{M}^1$ are kept small to prevent overfitting to the fixed global structure: $\mathcal{M}^1$ thus uses a receptive field of 5^3, while the rest use 9^3. We train our diffusion models with $T=1000$ diffusion steps. For sampling, we set $T[1]=1000$ and $T[l>1]=300$. $\mathcal{M}^1$ is trained for 20,000 iterations, and the rest for 40,000 iterations. All levels are trained with random crops of the same resolution to help ensure that each scale is trained in roughly the same time, and we use a learning rate of 10^{-4} with a 1% dropout rate.

Feature Extraction. In terms of shape processing, we normalize each mesh to fit within $[-1,1]^3$. 3D features are sampled at a resolution of 1024^3 and downsampled to a corasest resolution of 16^3 voxels, as described in Sec. 3.1.

5. Experiments

Data. We demonstrate our approach on 3D textured exemplars provided by Sin3DM (from [1, 4, 9, 19, 37, 47, 95]), and also used an open surface example that we created. Note that ShapeShifter can operate as-is on untextured inputs; but colors can help distinguish geometrically similar, yet semantically different parts of the geometry.

5.1. Comparison.

Baselines. We compare with two state-of-the-art papers on 3D generation from single examples: Sin3DM [107] and Sin3DGen [50]. Sin3DM uses a single-scale triplane diffusion model with a small receptive field to learn internal feature distribution within the exemplar shape. Features are learned in a separate autoencoder that parameterizes the input shape as an implicit neural surface [99]. We use their publicly available generated results for comparison. Sin3DGen operates instead on radiance field represented by plenoxels: it learns a hierarchical deformation field to transform the input plenoxels based on patch similarity. Following their data preparation guideline, we first rendered 200 images with Blender [7], then trained a 512^3 plenoxel to obtain the input exemplar which we provide to Sin3DGen to generate results to which we compare ourselves.

Quantitative Evaluations. Following prior work, we use Single Shape Fréchet Inception Distance (SSFID [106]) to evaluate the output quality, and pairwise (1−IoU)-distances among 10 generated variations to evaluate the output diversity. SSFID is extended from Single-Image Fréchet Inception Distance (SIFID) [87], which compares the statistics of the input and the generation feature extracted at different levels of a pretrained multiscale 3D encoder [16] trained on voxelized shapes from ShapeNet; thus we voxelized input and generations at a 256^3 resolution for evaluation purposes. We show in Tab. 1 that ShapeShifter outperforms competing methods in SSFID. (Note that our SSFID scores for Sin3DM differ from their reported scores as they used post-processed smoothed shapes as reference; please refer to the supplementary material for details.) While SSFIDs usually match the perceptual quality of the results, we note that the voxelization step used for scoring removes sharp features and high-frequency details, which our paper captures particularly well. The qualitative examples shown in Fig. 2 and in our supplementary material better demonstrate our strengths. Overall, the underlying geometry from Sin3DGen results is heavily distorted as it relies on plenoxel matching and blending, and while the SDF supervision from Sin3DM makes for better outputs, it does not capture sharp features well and the features extracted from their single-resolution triplane representation struggle to encode high-frequency details. Finally, the diversity scores based on IoU must be handled with care as it rewards artifacts. Our results show low diversity scores for very structured exemplars like the acropolis or the house, but good scores for more organic or varied shapes like the canyon or the small town, which is in line with our goal of generating variants of the input shapes.

Training and inference speed. We show timing comparisons in Tab. 2, where we also included GAN-based SSG [106] as its architecture is faster than its successor at inference time, albeit with lower quality and diversity [107]. Despite its impressive inference time, the training time of SSG is the longest. Sin3DGen does not require training as it is based on patch-matching; but each variation generation takes around 2 minutes, limiting interactive use cases.

Method	encoding and training	inference
SSG	4 hours	0.1 sec
Sin3DM	2.5 hrs	15.8 sec
Sin3DGen*	15 min	139 sec
ShapeShifter	12 min	10.7 sec

Table 2. Timings. Sin3DGen [50] was tested on a more performant GPU (Nvidia A100 40GB), as it could not fit in our regular GPUs (Nvidia 3080 10GB).

Sin3DM takes in total of 2.5 hours to train, of which ~30 minutes are used to learn triplane features. Our method takes merely 6 seconds to encode the shape, trains the 5-level hierarchy of diffusion models in 12 minutes, significantly outperforming all other trained models. The inference time takes from 0.15 seconds (at level 2) to 7.5 seconds (at level 5) totaling 10.7 seconds. It is worth mentioning that since our method outputs colored oriented pointset at every level, requiring no additional conversion e.g. marching cubes in other methods, we can flexibly choose the working level depending on the application. For example, in editing, we can operate on level 3, which takes only 1 s. per generation, thus enabling interactive modeling as shown next. More details are provided in the supplementary material.

5.2. Control and editing.

Our multiscale explicit representation makes it easy to control and edit the output. We demonstrate two examples: the user can trivially change the span of the model by resizing the initial grid $\mathbf{G}_T^0$ anisotropically, see Fig. 4; moreover, Fig. 5 demonstrates that a generated output can be further edit by copy-and-pasting parts of the output within one of the levels of the multiscale description of the shape, here to remove windows or adding a bay window. While existing works can offer similar capabilities, their use of triplane features or deformation fields to drive the generation renders editing less intuitive. For example, a patch from the input shape can be duplicated in Sin3DM to appear in the generated variations; however, the duplication must be done on three interdependent triplanes features, which do not directly correspond to a feature in 3D space.

5.3. Open Surfaces

Our use of points and normals to represent the geometry makes the treatment of open surfaces not only possible but just as simple as the case of closed surfaces: only the surface extraction method needs to be altered, i.e. with APSS [26] instead of [43]. An example is shown in Fig. 6.

5.4. Ablation studies

Learned Upsampler. We demonstrate the benefit of our learned upsampler. Replacing our upsampler (both in training and inference) by a trilinear interpolation as used in SinDDM produces visible artifacts (Fig. 7): our point features off-centered from the cell centers get entangled through trilinear upsampling.

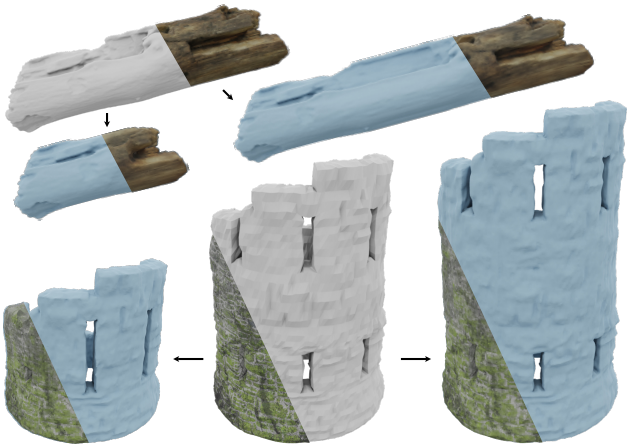


Figure 4. Controlled generation. The span of the output can be trivially controlled by resizing the initial grid anisotropically.

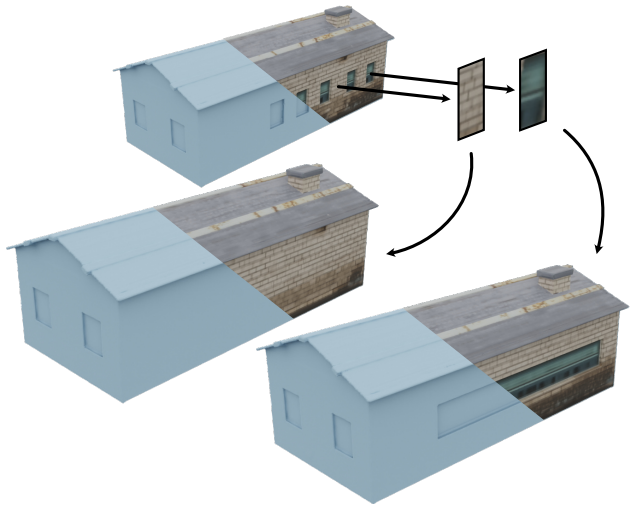


Figure 5. Editing. Using sparse voxel grid allows users to intuitively apply more precise edits. Here, a user can copy and paste a selected part of a generated variation at an intermediate level to manually alter the variation.

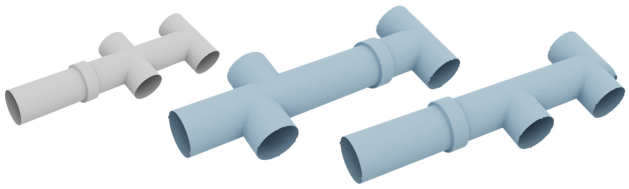
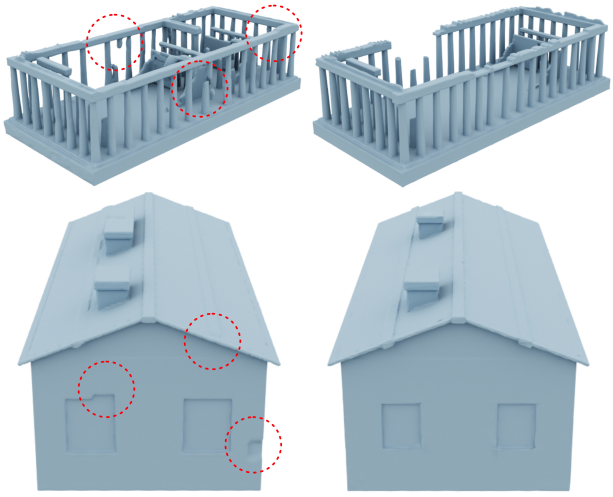


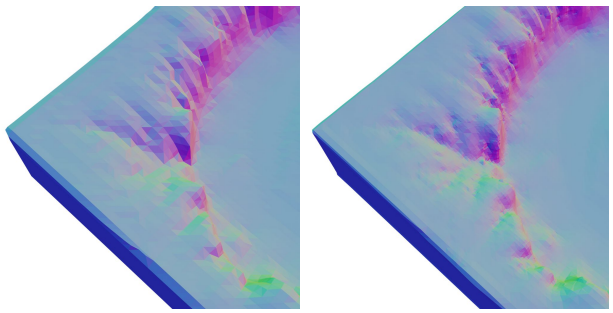
Figure 6. Open surfaces. Our oriented points representation also handles open surfaces by simply using APSS [26] to mesh the generated point set, while it is challenging for existing methods.

3D Features. We also replace our geometric features with an SDF. For fairness of comparison, we use two layers of active cells (instead of one) close to the surface to compensate for the reduced feature dimensionality. For the same input resolution, our features have more details (Fig. 8).



(a) Trilinear Interpolation (b) Learned Upsampler

Figure 7. Ablation test of upsampling. Comparing trilinear interpolation (left) with learned upsampler (right), the interpolation causes artifacts (see circled areas), whereas the learned upsampler provides a more detailed and structurally coherent output.



(a) SDF + Marching Cube [60] (b) Ours + Poisson [43]

Figure 8. Ablation test of features. Comparing SDF (left) with our proposed point and normal features (right) at the same resolution (128^3) demonstrates that our proposed features produce richer geometric details. The mesh color encodes the normal direction to reflect the difference in geometry details.

5.5. Texture Augmentation

Finally, we show that one can texture our generated models by applying contemporary image super-resolution on the baked texture maps in Fig. 9: using Magnific AI [63] for example can efficiently generate a fine texture improving the visual impact of our results. While this is only a proof-of-concept example, exploring the texturing of our geometric models is an exciting, albeit orthogonal, research direction.

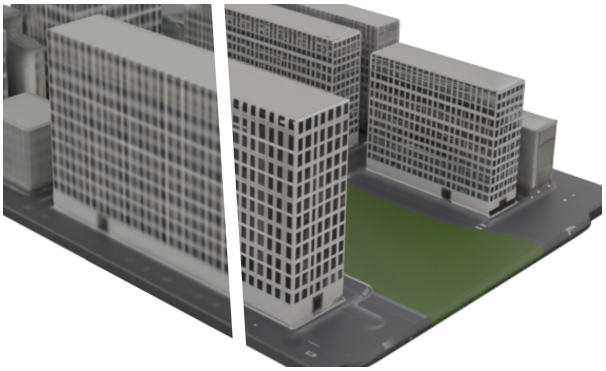


Figure 9. Texturing. As explained Sec. 3.3, ultra-high resolution texture can be obtained by applying state-of-the-art AI image-enhancing tools on the texture maps created by our method from the colored point set outputs.

6. Limitations and Future Work

Just like previous exemplar-based generative methods, ShapeShifter is limited in the shape variations it can generate: while our approach is not strictly patch-based, it is similarly restricted in its ability to consider widely different variants. Extending its range of alterations through data augmentation or more involved (equivariant) features remains an intriguing possibility that would broaden the applicability of our method. Moreover, we focused our approach on generating high-quality, detailed geometry and did not consider fine texture generation. While existing exemplar-based methods have proposed approaches to generate textures for meshes that we could apply as-is, we believe there may be other exciting possibilities to explore, such as fitting 2D Gaussian splats [35] within our finest voxels to enrich our geometry with radiance field reconstruction.

Now that we have proven the efficacy of explicit geometry encoding through colored points and normals for creating shapes in our exemplar context, it would be interesting to study its adequacy in the more general case of generative modeling from large datasets: its lightweight, surface-based nature may circumvent a number of issues plaguing current state-of-the-art approaches.

7. Conclusion

We proposed a novel generative approach for generating high-quality and detailed 3D models from a single exemplar. Our approach stands out as the first 3D generative method based on an explicit encoding of geometry through points, normals, and optionally colors. Combined with sparse voxel grid, we demonstrated that both training and inference times are (at times drastically) reduced compared to previous methods, despite a significantly improved quality of our geometric outputs and an ability to deal seamlessly with closed or open surfaces alike. We thus believe

that ShapeShifter sets a new standard for the quality of geometric outputs in generative modeling.

References

[1] All about Blender-3D. Photo-realistic floating wood. https://www.cgtrader.com/free-3d-models/plant/other/photo-realistic-floating-wood, 2020. License: Royalty Free. 6

[2] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In International conference on machine learning, pages 40–49. PMLR, 2018. 2

[3] and others. State of the art on diffusion models for visual computing. In Computer Graphics Forum, page e15063. Wiley Online Library, 2024. 2

[4] Pedram Ashoori. Small town. https://www.cgtrader.com/free-3d-models/exterior/cityscape/small-town-87b127c8-c991-4063-aa69-e58800686299, 2020. License: Royalty Free. 6

[5] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. ACM Trans. Graph., 28(3):24, 2009. 3

[6] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks, 5(2):157–166, 1994. 2

[7] Blender Online Community. Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 6

[8] Tianshi Cao, Karsten Kreis, Sanja Fidler, Nicholas Sharp, and Kangxue Yin. Texfusion: Synthesizing 3d textures with text-guided image diffusion models. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 4169–4181, 2023. 4

[9] Lukas Carnota. Industrial building. https://www.cgtrader.com/free-3d-models/exterior/office/indusrtial-building, 2015. License: Royalty Free. 6

[10] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 16123–16133, 2022. 2, 3

[11] Hansheng Chen, Jiatao Gu, Anpei Chen, Wei Tian, Zhuowen Tu, Lingjie Liu, and Hao Su. Single-stage diffusion nerf: A unified approach to 3d generation and reconstruction. In Proceedings of the IEEE/CVF international conference on computer vision, pages 2416–2425, 2023. 3

[12] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In Proceedings of the IEEE/CVF international conference on computer vision, pages 22246–22256, 2023. 3

[13] Ting Chen. On the Importance of Noise Scheduling for Diffusion Models, 2023. arXiv:2301.10972. 4

[14] Yiwen Chen, Tong He, Di Huang, Weicai Ye, Sijin Chen, Jiaxiang Tang, Xin Chen, Zhongang Cai, Lei Yang, Gang Yu, et al. Meshanything: Artist-created mesh generation with autoregressive transformers. arXiv preprint arXiv:2406.10163, 2024. 2

[15] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 3

[16] Zhiqin Chen, Vladimir G Kim, Matthew Fisher, Noam Aigerman, Hao Zhang, and Siddhartha Chaudhuri. Decor-gan: 3d shape detailization by conditional refinement. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 15740–15749, 2021. 6

[17] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander G Schwing, and Liang-Yan Gui. Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4456–4465, 2023. 3

[18] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. Advances in Neural Information Processing Systems, 36, 2024. 2

[19] DJMaesen. Cliff. https://sketchfab.com/3d-models/cliff-082da1166a814c6e9c9e6c1b38159e4e, 2021. License: CC Attribution. 6

[20] Ariel Elnekave and Yair Weiss. Generating natural images with direct patch distributions matching. In European Conference on Computer Vision, pages 544–560. Springer, 2022. 3

[21] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 5501–5510, 2022. 3

[22] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. Advances In Neural Information Processing Systems, 35:31841–31854, 2022. 2

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. Communications of the ACM, 63(11):139–144, 2020. 2

[24] Niv Granot, Ben Feinstein, Assaf Shocher, Shai Bagon, and Michal Irani. Drop the gan: In defense of patches nearest neighbors as single image generative models. In Proceedings of the IEEE/CVF conference on computer

vision and pattern recognition, pages 13460–13469, 2022. 3

[25] Alex Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013. 2

[26] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. In ACM SIGGRAPH 2007 papers, page 23, San Diego California, 2007. ACM. 5, 7

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

[28] Niv Haim, Ben Feinstein, Niv Granot, Assaf Shocher, Shai Bagon, Tali Dekel, and Michal Irani. Diverse generation from a single video made possible. In European Conference on Computer Vision, pages 491–509. Springer, 2022. 3

[29] Charles Han, Eric Risser, Ravi Ramamoorthi, and Eitan Grinspun. Multiscale texture synthesis. In ACM SIGGRAPH 2008 papers, pages 1–8. 2008. 3

[30] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. ACM Transactions on Graphics (ToG), 38(4):1–12, 2019. 3

[31] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. Deep geometric texture synthesis. ACM Trans. Graph., 39(4), 2020. 3

[32] Tobias Hinz, Matthew Fisher, Oliver Wang, and Stefan Wermter. Improved techniques for training single-image gans. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 1300–1309, 2021. 3

[33] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. Advances in neural information processing systems, 33:6840–6851, 2020. 2, 5

[34] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: Large reconstruction model for single image to 3d. In The Twelfth International Conference on Learning Representations, 2024. 3

[35] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In ACM SIGGRAPH 2024 Conference Papers, 2024. 8

[36] Dong Huo, Zixin Guo, Xinxin Zuo, Zhihao Shi, Juwei Lu, Peng Dai, Songcen Xu, Li Cheng, and Yee-Hong Yang. Texgen: Text-guided 3d texture generation with multi-view sampling and resampling. In European Conference on Computer Vision, pages 352–368. Springer, 2025. 3

[37] ImpJive. Fighting pillar. https://sketchfab.com/3d-models/fighting-pillar-14e73d2d9e8a4981b49d0e6c56d30af5, 2021. License: CC Attribution-ShareAlike. 6

[38] Alec Jacobson and Daniele Panozzo. libigl, 2023. 5

[39] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. arXiv preprint arXiv:2305.02463, 2023. 3

[40] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10124–10134, 2023. 4

[41] Animesh Karnewar, Oliver Wang, Tobias Ritschel, and Niloy J Mitra. 3ingan: Learning a 3d generative model from images of a self-similar scene. In 2022 International Conference on 3D Vision (3DV), pages 342–352. IEEE, 2022. 3

[42] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3907–3916, 2018. 3

[43] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. ACM Transactions on Graphics, 32 (3):1–13, 2013. 5, 7, 8

[44] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graph., 42(4):139–1, 2023. 3

[45] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. Foundations and Trends® in Machine Learning, 12(4):307–392, 2019. 2

[46] Vladimir Kulikov, Shahar Yadin, Matan Kleiner, and Tomer Michaeli. Sinddm: A single image denoising diffusion model. In International conference on machine learning, pages 17920–17930. PMLR, 2023. 3, 4

[47] Choly Kurd. Akropolis. https://www.turbosquid.com/3d-models/acropolis-3ds-free/610885, 2021. License: Educational Uses. 6

[48] Haoying Li, Yifan Yang, Meng Chang, Shiqi Chen, Huajun Feng, Zhihai Xu, Qi Li, and Yueting Chen. Srdiff: Single image super-resolution with diffusion probabilistic models. Neurocomputing, 479:47–59, 2022. 4

[49] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. In The Twelfth International Conference on Learning Representations, 2024. 3

[50] Weiyu Li, Xuelin Chen, Jue Wang, and Baoquan Chen. Patch-based 3d natural scene generation from a single example. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16762–16772, 2023. 1, 2, 3, 6, 7

[51] Weiyu Li, Rui Chen, Xuelin Chen, and Ping Tan. Sweetdreamer: Aligning geometric priors in 2d diffusion for consistent text-to-3d. In The Twelfth International Conference on Learning Representations, 2024. 3

[52] Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6517–6526, 2024. 3

[53] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja

Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 300–309, 2023. 3

[54] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. Advances in Neural Information Processing Systems, 36, 2024. 3

[55] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In Proceedings of the IEEE/CVF international conference on computer vision, pages 9298–9309, 2023. 3

[56] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In Proceedings of the IEEE/CVF international conference on computer vision, pages 7708–7717, 2019. 3

[57] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. In The Twelfth International Conference on Learning Representations, 2024. 3

[58] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: learning dynamic renderable volumes from images. ACM Transactions on Graphics (TOG), 38(4):1–14, 2019. 3

[59] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9970–9980, 2024. 3

[60] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In Seminal graphics: pioneering efforts that shaped the field, pages 347–353. 1998. 8

[61] Artem Lukoianov, Haitz Sáez de Ocáriz Borde, Kristjan Greenewald, Vitor Campagnolo Guizilini, Timur Bagautdinov, Vincent Sitzmann, and Justin Solomon. Score distillation via reparametrized ddim. arXiv preprint arXiv:2405.15891, 2024. 3

[62] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 2837–2845, 2021. 3

[63] Magnific AI. Magnific ai: Accelerating scientific research. https://magnific.ai/. Accessed: 2024-11-13. 8

[64] Nissim Maruani, Roman Klokov, Maks Ovsjanikov, Pierre Alliez, and Mathieu Desbrun. VoroMesh: Learning Watertight Surface Meshes with Voronoi Diagrams. In 2023 IEEE/CVF International Conference on Computer Vision (ICCV), pages 14519–14528, Paris, France, 2023. IEEE. 3

[65] Nissim Maruani, Maks Ovsjanikov, Pierre Alliez, and Mathieu Desbrun. PoNQ: A Neural QEM-Based Mesh Representation. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 3647–3657, Seattle, WA, USA, 2024. IEEE. 3, 4, 2

[66] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 4460–4470, 2019. 3

[67] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12663–12673, 2023. 3

[68] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM, 65(1):99–106, 2021. 3

[69] Thomas W. Mitchel, Carlos Esteves, and Ameesh Makadia. Single mesh diffusion models with field latents for texture generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 7953–7963, 2024. 3

[70] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3d meshes. In International conference on machine learning, pages 7220–7229. PMLR, 2020. 2

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

[72] Yaniv Nikankin, Niv Haim, and Michal Irani. Sinfusion: training diffusion models on a single image or video. In Proceedings of the 40th International Conference on Machine Learning, pages 26199–26214, 2023. 3, 4

[73] Eyvind Niklasson, Alexander Mordvintsev, Ettore Randazzo, and Michael Levin. Self-organising textures. Distill, 6(2):e00027–003, 2021. 3

[74] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 165–174, 2019. 2, 3

[75] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, 2019. arXiv:1912.01703 [cs, stat]. 5

[76] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In The Eleventh International Conference on Learning Representations, 2023. 3

[77] Sergey Prokudin, Qianli Ma, Maxime Raafat, Julien Valentin, and Siyu Tang. Dynamic Point Fields. In 2023

IEEE/CVF International Conference on Computer Vision (ICCV), pages 7964–7976, 2023. 2

[78] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 652–660, 2017. 3

[79] Lingteng Qiu, Guanying Chen, Xiaodong Gu, Qi Zuo, Mutian Xu, Yushuang Wu, Weihao Yuan, Zilong Dong, Liefeng Bo, and Xiaoguang Han. Richdreamer: A generalizable normal-depth diffusion model for detail richness in text-to-3d. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9914–9925, 2024. 3

[80] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. arXiv:2007.08501, 2020. 2

[81] Xuanchi Ren, Jiahui Huang, Xiaohui Zeng, Ken Museth, Sanja Fidler, and Francis Williams. Xcube: Large-scale 3d generative modeling using sparse voxel hierarchies. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4209–4219, 2024. 3

[82] Xuanchi Ren, Yifan Lu, Hanxue Liang, Jay Zhangjie Wu, Huan Ling, Mike Chen, Francis Fidler, Sanja annd Williams, and Jiahui Huang. Scube: Instant large-scale scene reconstruction using voxsplats. In The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024. 3

[83] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In International conference on machine learning, pages 1530–1538. PMLR, 2015. 2

[84] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. In ACM SIGGRAPH 2023 conference proceedings, pages 1–11, 2023. 2, 4

[85] Carlos Rodriguez-Pardo and Elena Garces. Seamlessgan: Self-supervised synthesis of tileable texture maps. IEEE Transactions on Visualization and Computer Graphics, 29 (6):2914–2925, 2022. 3

[86] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. Advances in Neural Information Processing Systems, 33:20154–20166, 2020. 3

[87] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In Proceedings of the IEEE/CVF international conference on computer vision, pages 4570–4580, 2019. 3, 4, 6, 1

[88] Yichun Shi, Peng Wang, Jianglong Ye, Long Mai, Kejie Li, and Xiao Yang. MVDream: Multi-view diffusion for 3d generation. In The Twelfth International Conference on Learning Representations, 2024. 3

[89] Assaf Shocher, Shai Bagon, Phillip Isola, and Michal Irani. Ingan: Capturing and retargeting the" dna" of a natural image. In Proceedings of the IEEE/CVF international conference on computer vision, pages 4492–4501, 2019. 3

[90] Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. Meshgpt: Generating triangle meshes with decoder-only transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 19615–19625, 2024. 2

[91] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In International conference on machine learning, pages 2256–2265. PMLR, 2015. 2

[92] Minjung Son, Jeong Joon Park, Leonidas Guibas, and Gordon Wetzstein. Singraf: Learning a 3d generative radiance field for a single scene. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 8507–8517, 2023. 1, 2, 3

[93] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In International Conference on Learning Representations, 2021. 5

[94] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In European Conference on Computer Vision, pages 1–18. Springer, 2025. 3

[95] Simon Ustal. Canyon landscape. https://sketchfab.com/3d-models/canyon-landscape-c395e9eb54ba4f40820ccfb98d3c2832, 2020. License: CC Attribution. 6

[96] Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, Karsten Kreis, et al. Lion: Latent point diffusion models for 3d shape generation. Advances in Neural Information Processing Systems, 35:10021–10039, 2022. 3

[97] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In International conference on machine learning, pages 1747–1756. PMLR, 2016. 2

[98] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12619–12629, 2023. 3

[99] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689, 2021. 6

[100] Peng Wang, Hao Tan, Sai Bi, Yinghao Xu, Fujun Luan, Kalyan Sunkavalli, Wenping Wang, Zexiang Xu, and Kai Zhang. PF-LRM: Pose-free large reconstruction model for joint pose and shape prediction. In The Twelfth International Conference on Learning Representations, 2024. 3

[101] Peihao Wang, Dejia Xu, Zhiwen Fan, Dilin Wang, Sreyas Mohan, Forrest Iandola, Rakesh Ranjan, Yilei Li, Qiang Liu, Zhangyang Wang, et al. Taming mode collapse in score distillation for text-to-3d generation. In Proceedings of

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9037–9047, 2024. 3

[102] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. Advances in Neural Information Processing Systems, 36, 2024. 3

[103] Xinyue Wei, Kai Zhang, Sai Bi, Hao Tan, Fujun Luan, Valentin Deschaintre, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Meshlrm: Large reconstruction model for high-quality mesh. arXiv preprint arXiv:2404.12385, 2024. 3

[104] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 7467–7477, 2020. 3

[105] Francis Williams, Jiahui Huang, Jonathan Swartz, Gergely Klar, Vijay Thakkar, Matthew Cong, Xuanchi Ren, Ruilong Li, Clement Fuji-Tsang, Sanja Fidler, et al. fvdb: A deep-learning framework for sparse, large scale, and high performance spatial intelligence. ACM Transactions on Graphics (TOG), 43(4):1–15, 2024. 2, 3, 4

[106] Rundi Wu and Changxi Zheng. Learning to generate 3d shapes from a single example. ACM Transactions on Graphics (TOG), 41(6):1–19, 2022. 1, 2, 3, 6

[107] Rundi Wu, Ruoshi Liu, Carl Vondrick, and Changxi Zheng. Sin3DM: Learning a diffusion model from a single 3d textured shape. In The Twelfth International Conference on Learning Representations, 2024. 1, 2, 3, 6

[108] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1912–1920, 2015. 3

[109] Yinghao Xu, Zifan Shi, Wang Yifan, Sida Peng, Ceyuan Yang, Yujun Shen, and Wetzstein Gordon. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. arxiv: 2403.14621, 2024. 3

[110] Yinghao Xu, Hao Tan, Fujun Luan, Sai Bi, Peng Wang, Jiahao Li, Zifan Shi, Kalyan Sunkavalli, Gordon Wetzstein, Zexiang Xu, and Kai Zhang. DMV3d: Denoising multi-view diffusion using 3d large reconstruction model. In The Twelfth International Conference on Learning Representations, 2024. 3

[111] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In Proceedings of the IEEE/CVF international conference on computer vision, pages 4541–4550, 2019. 2

[112] Haibo Yang, Yang Chen, Yingwei Pan, Ting Yao, Zhineng Chen, Zuxuan Wu, Yu-Gang Jiang, and Tao Mei. Dreammesh: Jointly manipulating and texturing triangle meshes for text-to-3d generation. arXiv preprint arXiv:2409.07454, 2024. 3

[113] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. ACM Transactions on Graphics (TOG), 38(6):1–14, 2019. 3

[114] Fukun Yin, Xin Chen, Chi Zhang, Biao Jiang, Zibo Zhao, Jiayuan Fan, Gang Yu, Taihao Li, and Tao Chen. Shapegpt: 3d shape generation with a unified multi-modal language model. arXiv preprint arXiv:2311.17618, 2023. 2

[115] Xianfang Zeng, Xin Chen, Zhongqi Qi, Wen Liu, Zibo Zhao, Zhibin Wang, Bin Fu, Yong Liu, and Gang Yu. Paint3d: Paint anything 3d with lighting-less texture diffusion models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4252–4262, 2024. 3

[116] Biao Zhang, Matthias Nießner, and Peter Wonka. 3dilg: Irregular latent grids for 3d generative modeling. Advances in Neural Information Processing Systems, 35:21871–21885, 2022. 2

[117] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. ACM Transactions on Graphics (TOG), 42(4):1–16, 2023. 3

[118] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In European Conference on Computer Vision, pages 1–19. Springer, 2025. 3

[119] Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. Clay: A controllable large-scale generative model for creating high-quality 3d assets. ACM Transactions on Graphics (TOG), 43(4):1–20, 2024. 3, 1

[120] Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. Advances in Neural Information Processing Systems, 36, 2024. 3

[121] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. arXiv preprint arXiv:1801.09847, 2018. 5

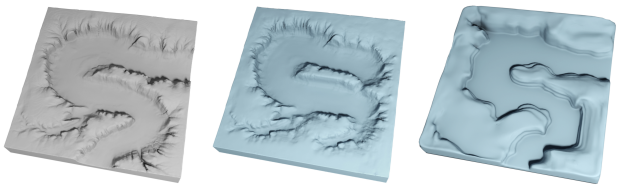
[122] Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Non-stationary texture synthesis by adversarial expansion. ACM Transactions on Graphics (TOG), 37(4):1–13, 2018. 3

[123] Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10324–10335, 2024. 3

ShapeShifter: 3D Variations Using Multiscale and Sparse Point-Voxel Diffusion

Supplementary Material

This supplementary material provides complement details, results, and comparisons.



(a) Input Geometry (b) Ours (c) Rodin

Figure 10. **ShapeShifter.** Given a 3D exemplar (left), we train a hierarchical diffusion model to create novel variations that preserve the geometric details and styles of the exemplar (center), whereas a large generative model such as Rodin [119] tends to lose the geometric details present in the input (right).

8. Additional results and renderings

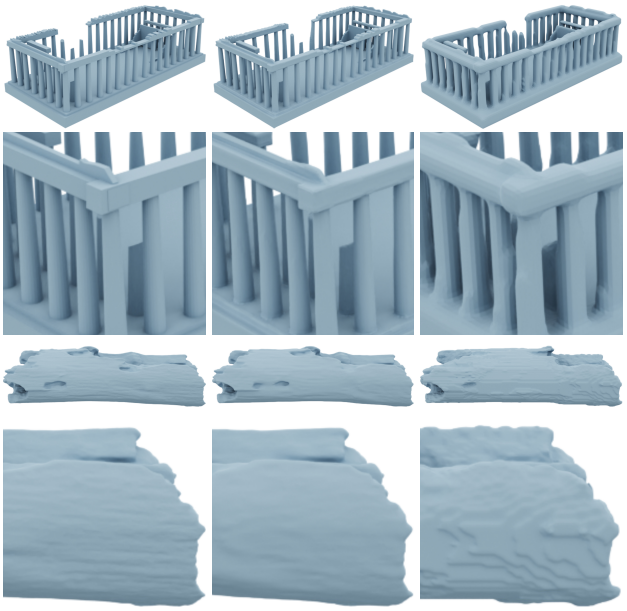
We provide more results in Fig. 10, 13, and 14 to better illustrate the outputs of ShapeShifter on a variety of reference models (including new ones compared to the submission). Note that we also show that ShapeShifter can generate purely geometric variants from untextured meshes (see last result in Fig. 14).

8.1. Comparison to SSG

We provide additional comparison with SSG [106], which is a 3D generalization of SinGAN [87] trained on multiscale triplane occupancy fields. Tab. 3 shows the quantitative evaluation on the models for which SSG provides publicly available outputs, demonstrating the higher quality of ShapeShifter. Furthermore, we demonstrate in Fig. 11 that the typical results of this GAN-based method exhibit the exaggerated smoothness like in all existing techniques, and often suffer from voxelized artifacts as well.

Metric	Method	acropolis	house	small-town	wood
G-Qual. ↓	SSG	2.81	0.91	1.71	0.07
	ShapeShifter	**0.01**	**0.01**	**1.00**	**0.02**
G-Div. ↑	SSG	**0.081**	**0.01**	0.19	**0.11**
	ShapeShifter	0.04	**0.01**	**0.60**	0.08

Table 3. **Evaluating geometric quality and diversity using SSFID and pairwise IoU scores.** As we discussed in our submission and in Sec. 9, both metrics have their blindspots: SSFID tends to overlook geometric details while pairwise IoU systematically rewards artifacts.



(a) Input Geometry (b) ShapeShifter (c) SSG

Figure 11. **Visual inspection of SSG [106] results.** While SSG can generate 3D outputs with very fast inference time, results are typically blobby or overly smooth, with spurious artifacts often visible due to its voxel-based generation process. In contrast, our method generates better sharp edges and subtle details.

8.2. Data-intensive vs. exemplar-based 3D generation

In this section, we discuss the value of exemplar-based 3D generations in light of the recent advancements in 3D generation models trained on millions of examples. The latter can be used to create highly diverse 3D assets and provide intuitive user controls through simple text and images. However, such models require immense computational resources for training and inference. Yet, as shown in Fig. 10, the state-of-the-art generator Rodin [119] (1.5B parameters) fails to create convincing geometric details comparable to those generated by our model.

Furthermore, the control provided by such models is limited, as the generation can only adhere to extremely coarse guidance. For example, in Fig. 10, we use the exemplar mesh as part of the inputs to Rodin for a conditioned generation. However, the output (right) completely loses the styles and details present in the exemplar mesh.

9. Additional comments on metrics

While we use the two commonly-used metrics (geometric quality and diversity through SSFID and pairwise IoU scores) to evaluate our results and compare them to prior art, a few comments are in order.

First, the validity of these two scores is debatable. While geometric quality is arguably fair but cannot really gauge the diversity of the results, the measure of diversity itself is quite delicate to analyze. In a sense, the diversity score rewards noise, not just real diversity. For instance, ten grids of random binary values would get a diversity of 0.66, while ten grids of axis-aligned planes that are not overlapping would have a score of 1.0 — so a diversity score mixes different properties. This partial inadequacy of the score is the reason why we state in the submission that geometric quality and geometric diversity should really be considered together to infer the success of an approach. Moreover, we also point out that the diversity scores should be clearly smaller for very structured models (like the acropolis model) than for free-form or organic shapes; our results have scores in line with this expected behavior, which seems more meaningful than systematic high scores which would point to noise artifacts instead of good results.

Second, we wish to point out that our scores of Sin3DM [107] are different from the ones they publish. The reason is that Sin3DM applies a pre-processing step to make the input meshes watertight. This initial step systematically inflates small details and thin surfaces such as the roof of the house or the entablature of the acropolis, which negates many of the advantages of one-shot generative modeling: it degrades (at times severely) the input, losing the very reason why creating variants of a carefully-designed input model is highly sought after, i.e., the high-quality geometry of the exemplar. So we compared their results to the unprocessed input models, and did not re-train their neural network because we assumed that they made their best efforts to fit ground-truth shapes. So one should be aware that the low geometric quality scores we provide reflect both the degradations of the pre-processing step and of their SDF-based generative approach — again, to account for the real use of these generative approaches.

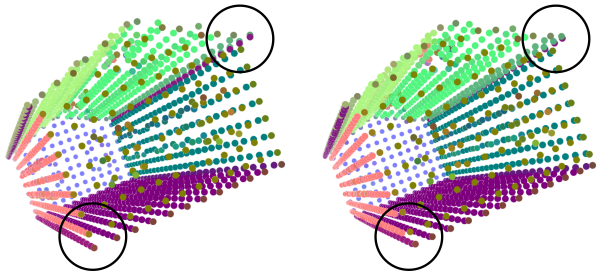
10. Inference timings

In the main paper, the inference time for ShapeShifter and Sin3DM is reported for generating 10 variants. Here, we provide the inference timing for generating a single variant (using a batch size of 1 instead of 10), as shown in Tab. 4.

Our method generates a single variant in less than 2 seconds: approximately 0.5 seconds for the coarsest level, followed by less than 1.5 seconds in total for the four finer levels. In comparison, Sin3DM requires around 5 seconds, while Sin3DGen takes over 3 minutes, excluding the time needed for optimizing the input plenoxels and converting them to a mesh. Notably, our method produces the coarsest level in under half a second, which can be directly splatted using [80] (see the video for live demonstrations). In contrast, Sin3DM[107] takes 5.18 seconds to process an equivalent grid size (32^3).

Method	Level 0	Level 1	Level 2	Level 3	Level 4
ShapeShifter	0.49	0.17	0.18	0.26	0.78
Sin3DM	-	5.18	-	-	-

Table 4. **Inference timing for generating a single variation.** We report the inference time at each level for generating a single variation and compare it with Sin3DM, which has a grid resolution equivalent as our second level (level 1), where as in the main paper we reported the inference time for 10 variations. DDIM sampling is used for both methods.

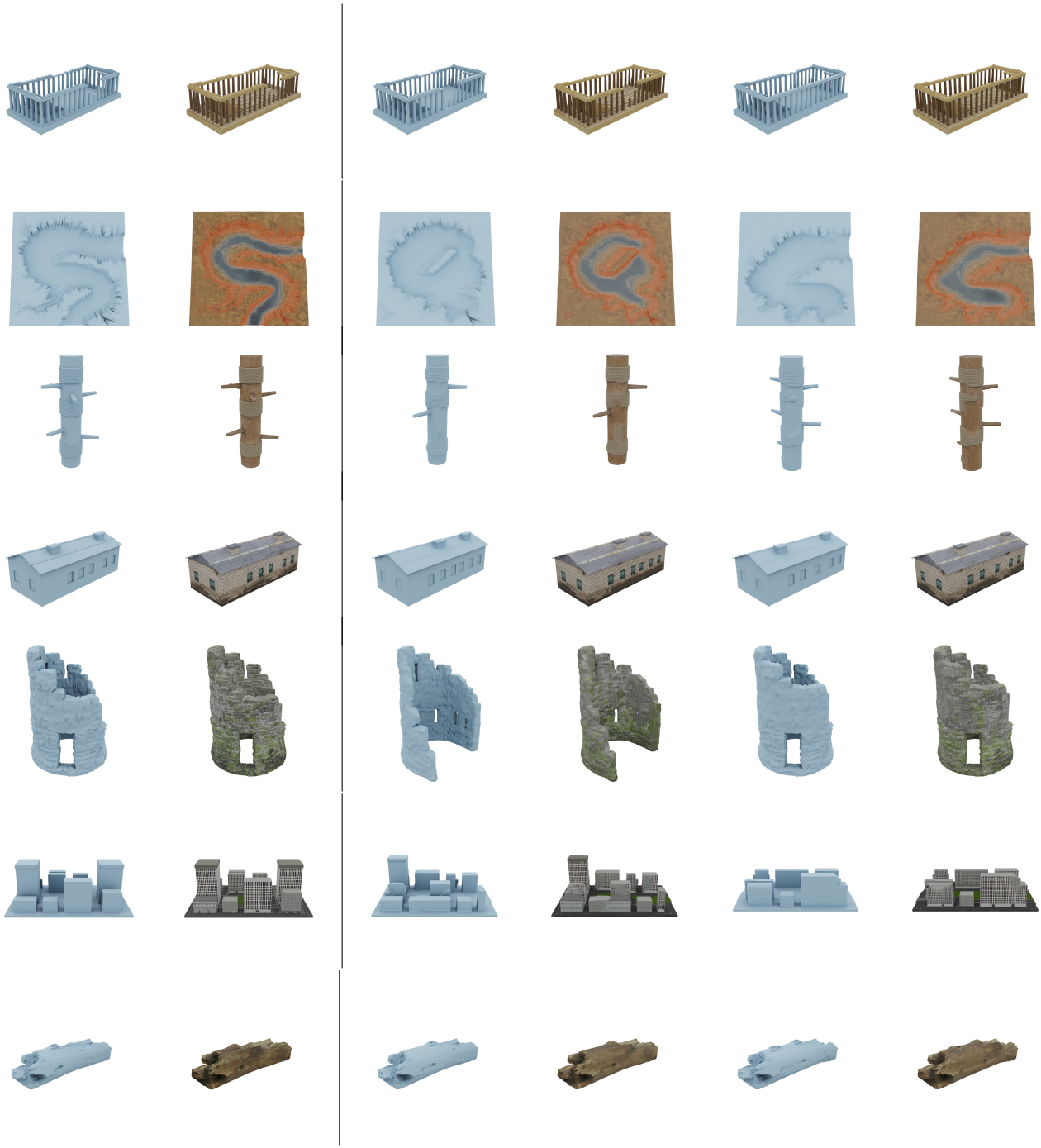


(a) QEM averaging (b) standard averaging

Figure 12. **QEM-averaging ablation.** While QEM-averaging (proposed in [65]) keeps sharp features (like corners or spikes) in place which helps our generative approach to maintain these local details, a usual averaging would move the "corner" points inwards, increasing the risk of smoothing features out in generated variants.

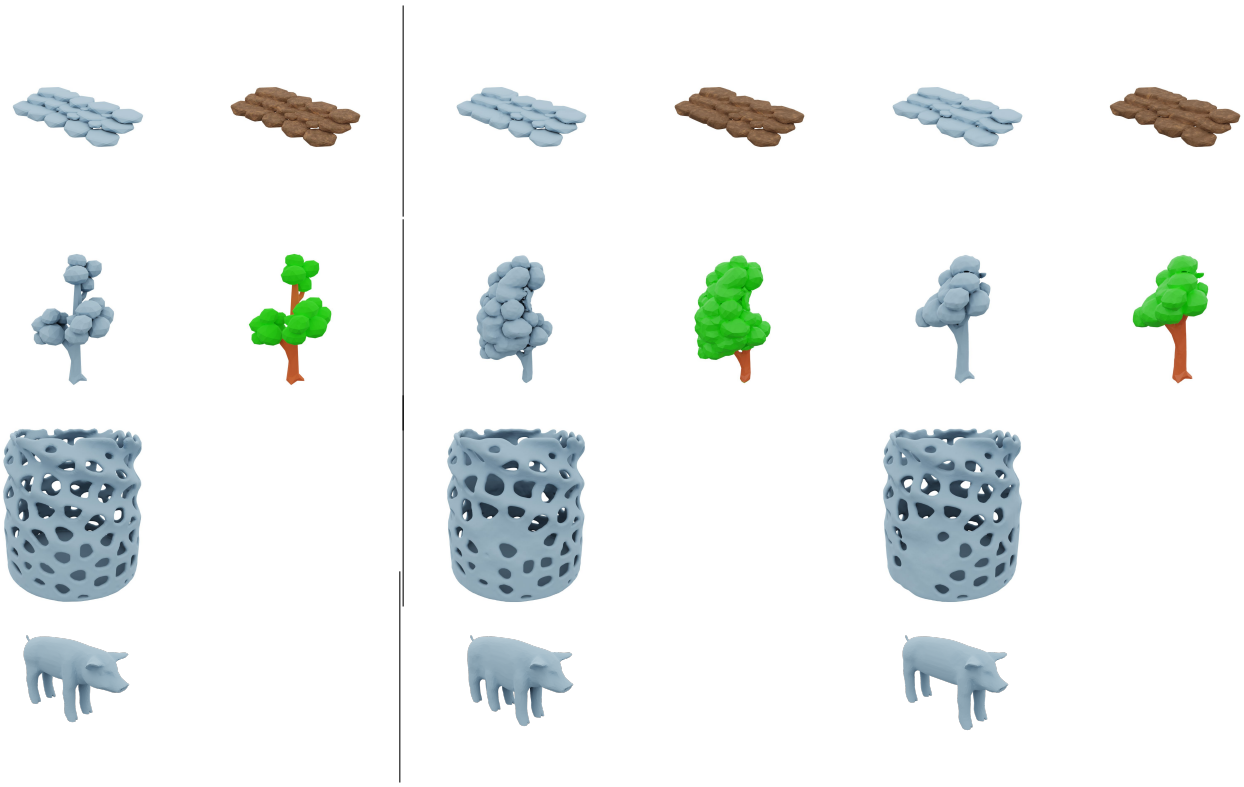
11. QEM averaging

Finally, we demonstrate why our use of QEM averaging during our fine-to-coarse analysis of the input models helps preserve sharp features of the ground truth. As Fig.12 demonstrates, standard scale-by-scale averaging of the points and normals from the finest sparse voxel grid all the way to the coarsest grid leads to drifts of the salient features: for instance, the bottom left corner of the house has migrated inwards, which may create rounding of the corner. Instead, applying the QEM averaging defined in the PoNQ method [65] allows for the placement of the coarsest point to stay on the corner, and of the intermediate points to remain right there as well — resulting in outputs which will better preserve this geometric feature.



(a) Input Shape (b) Generated Shape (ours) (c) Generated Shape (ours)

Figure 13. **Samples of our results I.** This figure shows a variety of input models and some of the generated variants (both shown without and with texture to facilitate visual inspection) ShapeShifter outputs.



(a) Input Shape (b) Generated Shape (ours) (c) Generated Shape (ours)

Figure 14. **Samples of our results II.** This figure shows input models that were not used in our submission, and some of the generated variants (both shown without and with texture to facilitate visual inspection) ShapeShifter outputs. Note that the last two examples (vase and pig) are an ablation test where we do not use colors among the per-voxel features in our approach.