$\mathcal{X}^3$: Large-Scale 3D Generative Modeling using Sparse Voxel Hierarchies

Xuanchi Ren[1,2,3] Jiahui Huang[1] Xiaohui Zeng[1,2,3] Ken Museth[1]
Sanja Fidler[1,2,3] Francis Williams[1]
[1]NVIDIA [2]University of Toronto [3]Vector Institute

Figure 1. **XCube ($\mathcal{X}^3$).** Our model generates high-resolution (up to 1024^3) sparse 3D voxel hierarchies of *objects* and *driving scenes* in under 30 seconds. The voxels are enriched with arbitrary attributes such as semantics, normals, and TSDF from which mesh could be readily extracted. Here we show randomly sampled geometries using our model trained on ShapeNet, Objaverse, Karton City, and Waymo.

Abstract

We present $\mathcal{X}^3$ (pronounced XCube), a novel generative model for high-resolution sparse 3D voxel grids with arbitrary attributes. Our model can generate millions of voxels with a finest effective resolution of up to 1024^3 in a feed-forward fashion without time-consuming test-time optimization. To achieve this, we employ a hierarchical voxel latent diffusion model which generates progressively higher resolution grids in a coarse-to-fine manner using a custom framework built on the highly efficient VDB data structure. Apart from generating high-resolution objects, we demonstrate the effectiveness of XCube on large outdoor scenes at scales of $100\,\mathrm{m} \times 100\,\mathrm{m}$ with a voxel size as small as $10\,\mathrm{cm}$. We observe clear qualitative and quantitative improvements over past approaches. In addition to unconditional generation, we show that our model can be used to solve a variety of tasks such as user-guided editing, scene completion from a single scan, and text-to-3D. More results and details can be found on our project webpage.

1. Introduction

Equipping machines with the ability to create and understand three-dimensional scenes and objects has long been a tantalizing pursuit, promising a bridge between the digital and physical worlds. The problem of 3D content generation lies at the heart of this endeavor. By modeling the distribution of objects and scenes, generative models can propose plausible shapes and their attributes from scratch, from user input, or from partial observations.

There has been a surge of new and exciting works on generative models for 3D shapes in recent years. Initial work in this area, train on datasets of 3D shapes and leverage 3D priors to perform generation [18, 22]. While these works produce impressive results, their diversity and shape quality is fundamentally bounded by the size of 3D datasets (*e.g.* [5, 12]), as well as their underlying 3D representation. To address the diversity problem, one line of work [47, 67] proposed an elegant solution that leverages powerful 2D generative models to produce 3D structures using inverse rendering and a diffusion score distillation formulation. While they benefit from the massive corpora of 2D image data and can generate highly diverse and high-quality shapes, the generated shapes usually suffer from the Janus face problem, and the generation process requires test-time optimization that is lengthy and computationally expensive. More recent works such as [31, 33] achieve state-of-the-art 3D generation by smartly combining a mix of 2D and 3D priors. They gain diversity from 2D data and spatial consistency from 3D data and speed up the generation process by operating directly in 3D. These works motivate a deeper investigation into the fundamentals of *3D priors* for generation. In particular, current 3D priors are limited to low resolutions, and do not scale well to large outdoor scenes such as those in autonomous

driving and robotics applications. These datasets of large-scale scenes are abundant and contain more data than those consisting solely of objects [60]. Thus, developing a scalable 3D generative prior has the potential to unlock new sources of training data and further push the boundary of what is possible in 3D generation. In this work, we aim to explore the limits of *purely 3D* generative priors, scaling them to high resolutions and large-scale scenes. Our model is capable of scaling to high-resolution outputs (e.g. 1024^3) by leveraging a novel sparse formulation and can produce outputs with high geometric complexity, by focusing dense geometry near the surface of a shape.

Our method, $\mathcal{X}^3$, is a novel hierarchical voxel latent diffusion model for generating high-resolution 3D objects and scenes with arbitrary attributes such as signed distances, normals, and semantics. Our model generates a latent *Sparse Voxel Hierarchy* — a hierarchy of 3D sparse voxel grids with latent features at each voxel — in a coarse-to-fine manner. In particular, we model each level of the hierarchy as a latent diffusion model conditioned on the coarser level. The latent space at each level is encoded using a highly efficient — both in terms of compute and memory — sparse structure Variational Autoencoder (VAE). Our generated representation enjoys several key benefits: (1) it is fully 3D, enabling it to model intricate details at multiple resolutions, (2) it can output very high-resolution shapes (up to 1024^3 resolution) by leveraging sparsity, (3) the distribution at each level is easy to model since the coarse level need only model a rough shape, and finer levels are concerned with local details, (4) our generated shapes support multi-scale user-guided editing by modifying coarser levels and regenerating finer levels, (5) since our model leverages a latent diffusion model over a hierarchy of features, we are able to decode *arbitrary* multi-scale attributes (*e.g.* semantics, TSDF) from those features.

We demonstrate the effectiveness of our hierarchical voxel latent diffusion model on standard object datasets such as Objaverse [12] and ShapeNet [5] achieving state-of-the-art results on unconditional and conditional generation from texts and category labels. We further demonstrate the scalability of our method by demonstrating high-quality unconditional and conditional (from a single lidar scan) generation on large outdoor scenes from the Waymo Open Dataset [60] and Karton City [1]. Finally, by leveraging a custom sparse 3D deep learning framework built on VDB [38], our model is capable of generating complex shapes at 1024^3 resolution containing *millions of voxels* in under 30 seconds.

2. Related Work

Generative Probabilistic Models. Common generative models include variational autoencoders (VAE) [27], generative adversarial networks (GAN) [17], normalizing flows [50], autoregressive models (AR) [66], and more recently diffusion models (DM) [19, 57]. A popular method for generative modeling is *latent diffusion* that has been found useful in, *e.g.*, images [46, 52, 65] and videos [3], where the diffusion process happens over the latent space of a simpler generative model (*e.g.* a VAE). Latent diffusion models allow for easy decoding of multiple attributes through different decoders. In our work, we employ a latent diffusion model over a hierarchy of sparse voxels.

3D Generative Models. The recent surge in the 3D generative modeling literature mostly focuses on *object-level* shape synthesis. One line of work opts for distilling 2D image priors into 3D via inverse rendering [29, 33, 67], while others [15, 24, 41, 43, 44, 71] focus on directly learning 3D priors from large-scale 3D datasets [11, 12]. Recently, hybrid 2D-3D approaches [32, 61] which better leverage both image priors and large 3D datasets have started to emerge. Fundamental to these works are good 3D priors that can instill multiview consistency without the need for expensive test-time optimization. This is the interest of our work.

Works that tackle large-scale *scene* generation either choose a procedural approach that decouples the generation of different scene components (*e.g.* roads, buildings, etc.) [49, 59, 69], or a more generic approach that generates the entire scene at once [8, 25, 30]. While the former approaches usually provide more details, they are limited to generating a fixed subset of possible scenes and require specialized data to train. The latter approaches are theoretically more flexible but are currently bounded by their 3D representation power (hence producing fewer details), which is a problem we address in this work.

3D Representation for Generative Tasks. Many popular 3D representations exist in the generative modeling literature. *Point clouds* [2, 35, 40, 71] are flexible and adaptive, but cannot represent solid surfaces and pose challenges in architecture design. *Triangle meshes* [16, 42] are more expressive but are limited to a fixed topology and hence hard to optimize. *Neural fields* [7, 34, 41] encode scene geometry implicitly in the network weights and lack an explicit inductive bias for effective distribution modeling. *Tri-planes* [6, 15, 56] can represent objects at high resolutions with reduced memory footprint, but fundamentally lack a geometric bias except for large axis-aligned planes, posing challenges when modelling larger scenes with complex geometry. Comparably, *voxel grids* [22, 55, 68] are flexible, expressive for both chunky and thin structures, and support fast querying and processing. *Sparse* voxel hierarchies do not store voxel information for empty regions and hence are more efficient. A popular approach in the literature is to implement these using octrees [23, 64] or hash tables [62]. However, previous works either focus only on geometry [64, 72], are limited to an effective resolution of 256^3 [23], do not consider hierarchical generation [73], or are not evaluated on large-scale, real-world datasets [28]. In contrast, our method can generate high-resolution shapes from a hierarchical latent space, and

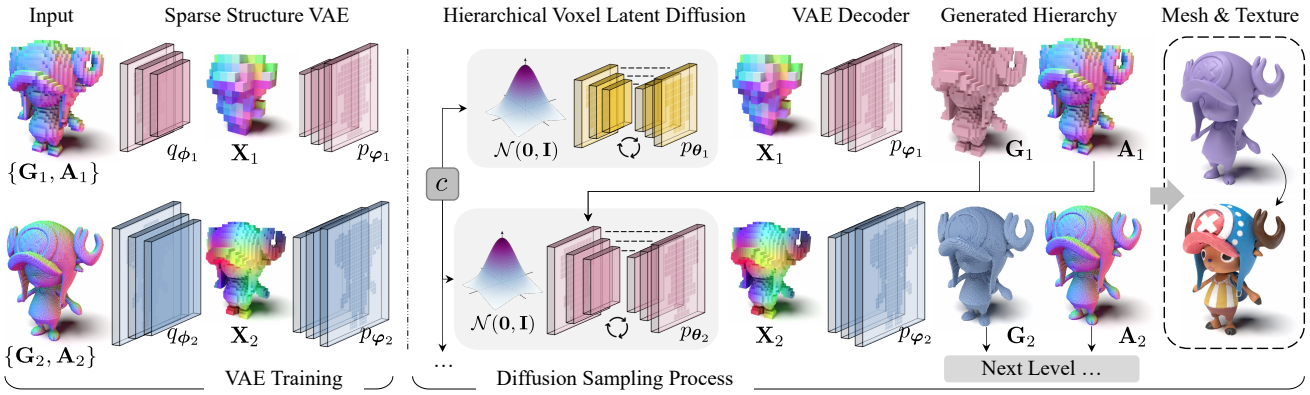


Figure 2. **Method.** Sparse voxel grids within the hierarchy are first encoded into compact latent representations using a sparse structure VAE. The hierarchical latent diffusion model then learns to generate each level of the latent representation conditioned on the coarser level in a cascaded fashion. The generated high-resolution voxel grids contain various attributes for different applications. Note that technically $\mathbf{X}_1$ is a dense latent grid, but illustrated as a sparse one for clarity.

is evaluated on large-scale, real-world scene datasets.

3. Method

Our goal is to learn a generative model of large-scale 3D scenes represented as sparse voxel hierarchies. The hierarchy comprises of L levels of coarse-to-fine voxel grids $\mathcal{G} = \{\mathbf{G}_1, ..., \mathbf{G}_L\}$ and their associated per-voxel attributes $\mathcal{A} = \{\mathbf{A}_1, ..., \mathbf{A}_L\}$ such as normals and semantics. Finer grids $\mathbf{G}_{l+1}$ with smaller voxel sizes are strictly contained within the coarser grids $\mathbf{G}_l$ for $l = 1, ..., L-1$, and the finest level of grid $\mathbf{G}_L$ contains the maximum amount of details.

Our method trains a hierarchy of latent diffusion models over the sparse voxel grids $\mathcal{G}$ encoded by a hierarchy of sparse structure VAEs, as summarized in Fig. 2. We first introduce the sparse structure variational autoencoder (VAE) that learns a compact latent representation of voxel grids in § 3.1. Then we describe our full diffusion probabilistic model that learns the joint distribution of the latent representation and the sparse voxel hierarchy in § 3.2. The training and sampling procedures are described in § 3.3, followed by the implementation details in § 3.4.

3.1. Sparse Structure VAE

Motivation. The sparse structure VAE is designed to learn a compact latent representation of each voxel grid within the hierarchy and its associated attributes. Instead of directly modeling their joint distribution that comprises a mixture of continuous and discrete random variables, we encode them into a unified continuous latent representation, which is not only friendly to the downstream diffusion models during training and sampling [65, 71], but also facilitates the formulation of a hierarchical probabilistic model which we aim to demonstrate. Additionally, the latent representation, encoded in a coarser spatial resolution, serves as a compact yet meaningful proxy that saves the computation while preserving the expressivity [18, 52]. Represented as $\mathcal{X} = \{\mathbf{X}_1, ..., \mathbf{X}_L\}$, the latent is also a featurized sparse voxel hierarchy corresponding to a coarser version of $\mathcal{G}$, with the voxel size of $\mathbf{X}_l$ being the same as $\mathbf{G}_{l-1}$.

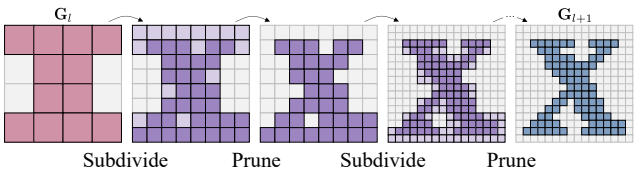


Figure 3. **VAE Decoder Architecture.** Coarser levels of grids $\mathbf{G}_l$ are upsampled to finer grids $\mathbf{G}_{l+1}$ by iteratively subdividing existing voxels into octants and pruning excessive ones. Each *level* may contain many upsampling *layers* that double the resolution.

Network Design. We choose to train separate VAEs that operate on each level l of the hierarchy independently. Hence for the ease of notation, we drop the subscript l in the following discussion. Here, we build the neural networks based on the operators over sparse voxel grids to model both the posterior distribution $q_\phi(\mathbf{X}|\mathbf{G}, \mathbf{A})$ and the likelihood distribution $p_\psi(\mathbf{G}, \mathbf{A}|\mathbf{X})$, with ϕ, ψ being the encoder and decoder weights respectively.

For the encoder, we utilize the sparse convolutional neural network to process the input $\mathbf{G}$ and $\mathbf{A}$ by alternatively applying sparse convolution and max pooling operations, downsampling to the resolution of $\mathbf{X}$. For the decoder, we borrow the structure prediction backbone from [21] that allows us to predict novel sparse voxels that are not present in the input. It starts from $\mathbf{X}$ and proceeds by progressively pruning excessive voxels and subdividing existing ones based on the prediction of a subdivision mask, and finally reaching the resolution of $\mathbf{G}$ after several upsampling layers. An illustration of the above decoding scheme is shown in Fig. 3.

3.2. Hierarchical Voxel Latent Diffusion

Probabilistic Modeling. Existing 3D generation literature [18, 41] typically uses one level of latent diffusion (*i.e.*

$L = 1$). While this is sufficient to generate intricate scenes containing one single object, the resolution is still far from enough to generate large-scale outdoor scenes. The limited scalability of their underlying 3D representation and the absence of probabilistic modeling to capture the coarse-to-fine nature of the data hinder the effectiveness of these methods. We solve this by marrying a hierarchical latent diffusion model [3, 20] with the sparse voxel representation. Specifically, we propose the following factorization of the joint distribution of grids and latents:

$$p(\mathcal{G}, \mathcal{A}, \mathcal{X}) = \prod_{l=1}^{L} p_{\boldsymbol{\psi}_l}(\mathbf{G}_l, \mathbf{A}_l | \mathbf{X}_l) p_{\boldsymbol{\theta}_l}(\mathbf{X}_l | \mathbf{C}_{l-1}), \quad (1)$$

where $\mathbf{C}_{l-1}$ is the condition from the coarser level, with:

$$\mathbf{C}_l = \begin{cases} c, & l = 0 \\ \{\mathbf{G}_l, \mathbf{A}_l, c\}, & l > 0 \end{cases}, \quad (2)$$

with c being an optional global condition such as a category label or a text prompt, and $p_{\boldsymbol{\theta}_l}(\cdot)$ instantiated as a diffusion model with parameter $\boldsymbol{\theta}_l$ which we elaborate on later.

The above factorization assumes the Markov process (*i.e.* level l is only conditioned on its coarser level $l-1$), which is naturally induced from the geometric nature of the data. By doing so, we reduce the layers in each level of VAE and amortize both the computation and the representation power across multiple levels (see § 4.4 for empirical proof). Additionally, such a factorized modeling endows us with utmost flexibility, enabling *user controls* by editing or resampling grids from different levels.

Diffusion Model $p_{\boldsymbol{\theta}}$. Here we omit subscript l again for clarity. A diffusion stochastic process transforms a complicated distribution of a random variable into the unit Gaussian distribution $\mathbf{X}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ by iteratively adding white noise to it, following a Markov process [19]. One commonly used instantiation is the following:

$$\mathbf{X}_t | \mathbf{X}_{t-1} \sim \mathcal{N}(\sqrt{1-\beta_t}\mathbf{X}_{t-1}, \beta_t \mathbf{I}), \quad (3)$$

where $0 < \beta_t \ll 1$ controls the amount of noise added for each step. The reverse process, on the other hand, removes the noise iteratively and reaches data distribution $\mathbf{X}_T$ within a discrete number of steps T. It is usually modeled as:

$$\mathbf{X}_{t-1} | \mathbf{X}_t \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{X}_t, t), \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \mathbf{I}), \quad (4)$$

with $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$, and $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ a parametrized learnable module. In practice, we re-parametrize $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ as:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}} = \sqrt{\alpha_t}\mathbf{X}_t - \beta_t \sqrt{\frac{\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}} \boldsymbol{v}_{\boldsymbol{\theta}}, \quad (5)$$

so that the learnable module predicts $\boldsymbol{v}$ instead. This is in accordance with the $\mathbf{v}$-parameterization in [53] that has been shown to facilitate training.

We instantiate $\boldsymbol{v}_{\boldsymbol{\theta}}(\cdot)$ as a 3D *sparse* variant of the backbone used in [13], ensuring the grid structure of the decoded output from $\boldsymbol{v}_{\boldsymbol{\theta}}$ matches the input. To inject condition $\mathbf{C}_{l-1}$, we directly concatenate the feature from $\mathbf{A}_{l-1}$ with the network input recalling that $\mathbf{X}_l$ also shares the same grid structure with $\mathbf{G}_{l-1}$. Timestep condition is implemented using AdaGN [52] and textual condition c is first CLIP-encoded [48] and then injected using cross attention.

3.3. Training and Sampling

Loss Functions. We train the VAE and the diffusion model level-by-level independently. During the training of the level-l VAE, we employ the following loss function:

$$\begin{aligned} \mathcal{L}_l^{\text{VAE}} = \mathbb{E}_{\{\mathbf{G}_l, \mathbf{A}_l\}}[\mathbb{E}_{\mathbf{X}_l \sim q_{\boldsymbol{\phi}}}[\text{BCE}(\mathbf{G}_l, \tilde{\mathbf{G}}_l) + \\ \mathcal{L}_l^{\text{Attr}}(\mathbf{A}_l, \tilde{\mathbf{A}}_l)] + \lambda\, \mathbb{KL}(q_{\boldsymbol{\phi}}(\mathbf{X}_l) \parallel p(\mathbf{X}_l))], \end{aligned} \quad (6)$$

where $\tilde{\mathbf{G}}_l, \tilde{\mathbf{A}}_l$ is the output of the VAE decoder $\boldsymbol{\psi}$ given $\mathbf{X}_l$, and $\mathcal{L}_l^{\text{Attr}}$ is the loss supervising the attribute predictions (*e.g.*, TSDF, semantics, etc.) with its specific form postponed to the supplementary material. $\text{BCE}(\cdot)$ is the binary cross entropy on the grid structure, making $p_{\boldsymbol{\psi}}$ a mixed product distribution. $\mathbb{KL}(\cdot \parallel \cdot)$ is the KL divergence between the posterior and the prior $p(\mathbf{X}_l)$, which we set to unit Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$, and λ is its weight.

The training loss for the diffusion model is:

$$\mathcal{L}_l^{\text{DM}} = \mathbb{E}_{t, \mathbf{X}_l, \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\| \boldsymbol{v}_{\boldsymbol{\theta}_l}(\mathbf{X}_{l,t}, t) - \boldsymbol{v}_{\text{ref}} \|_2^2 \right], \quad (7)$$

where $\boldsymbol{v}_{\text{ref}} = \sqrt{\bar{\alpha}_t}\boldsymbol{\varepsilon} - \sqrt{1-\bar{\alpha}_t}\mathbf{X}_l$, $t \sim [1, T]$, $\mathbf{X}_l$ is sampled from the VAE posterior, and $\mathbf{X}_{l,t} = \sqrt{\bar{\alpha}_t}\mathbf{X}_l + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}$.

Sampling. To sample from the joint distribution of Eq (1), one starts by drawing the coarest latent $\mathbf{X}_1$ from the diffusion model $p_{\boldsymbol{\theta}_1}$. Then, the decoder $p_{\boldsymbol{\psi}_1}$ is used to generate the coarsest grid $\mathbf{G}_1$ and its associated attributes $\mathbf{A}_1$ (which is then optionally refined by the refinement network). Conditioned on $\mathbf{C}_1 = \{\mathbf{G}_1, \mathbf{A}_1, c\}$, the diffusion model $p_{\boldsymbol{\theta}_2}$ is used to generate the next level of latent $\mathbf{X}_2$, and the process goes on until the highest resolution of $\{\mathbf{G}_L, \mathbf{A}_L\}$ is met. We include TSDF in $\mathbf{A}_L$ for all our experiments, which enables us to decode high-resolution meshes. For other tasks such as perception, we further allow for decoding other attributes such as semantics. We use DDIM [58] as our sampler.

3.4. Implementation Details

In practice, we find the following implementation details, specially tuned for the sparse voxel hierarchy generation case, to be helpful for better results: (1) **Early dilation**. In network layers with larger voxel sizes, we dilate the sparse voxel grids by one, so that the halo regions of the sparse topology also represent non-zero features. This helps later layers to better capture the local context and generate smooth structures. (2) **Refinement network**. An inherent problem

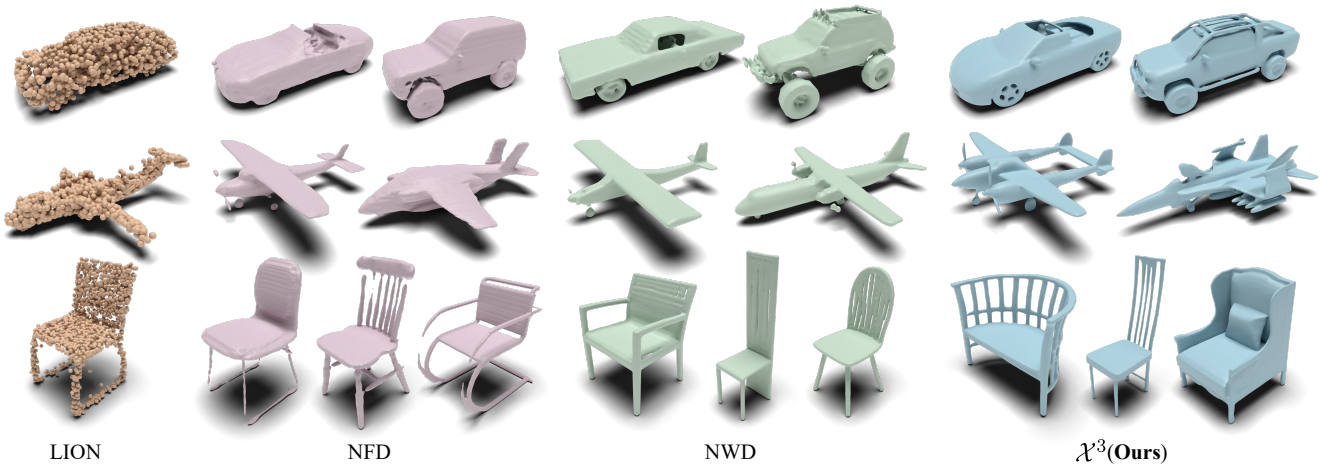


Figure 4. **ShapeNet [5] Qualitative Comparison.** We show comparison of our method with LION [71], NFD [56], and NWD [22]. Our method is capable of generating intricate geometry and thin structures. Best viewed with 200% zoom-in.

of our factorized modeling is error accumulation, where higher-resolution grids cannot easily fix the artifacts from prior layers. We mitigate this by appending a refinement network to the output of the VAE decoder that refines $\mathbf{G}_l$ and $\mathbf{A}_l$. The architecture of the refinement network is similar to [21], and its training data is augmented by adding noise to the posterior of the VAE [20] before being decoded. Last but not least, our architecture and training details can be found in the supplementary.

Sparse 3D Learning Framework. The use of sparse voxel grids motivates and enables us to build a custom 3D deep learning framework for sparse data in order to support higher spatial resolution and faster sampling. To this end, we leverage the VDB [38] structure to store our sparse 3D voxel grid. Thanks to its compact representation (taking only 11MB for 3.4 million of voxels) and fast look-up routine, we are able to implement common neural operators such as convolution and pooling in a very efficient manner. Fully operating on the GPU (including grid building), our framework is able to process a 3D scene with 1024^3 in milliseconds, runs $\sim 3\times$ faster with $\sim 0.5\times$ of the memory usage than the current state-of-the-art sparse 3D learning framework TorchSparse [62]. All our architectures are based on this custom framework.

4. Experiments

We conduct comprehensive experiments to evaluate the performance of our model. First, we demonstrate XCube's ability to perform unconditional object-level 3D generation using ShapeNet [5] (§ 4.1), and conditional 3D generation from category and text using Objaverse [12] (§ 4.2). Next, we showcase high-resolution outdoor scene-level 3D generation using both the Karton City [1] and Waymo [60] datasets (§ 4.3), which is one of the first results of this kind. Finally, we conduct ablation studies for our design choices (§ 4.4). Please refer to the supplementary for further results.

	Airplane		Chair		Car	
	CD	EMD	CD	EMD	CD	EMD
Point-based method						
PVD [74]	69.55	60.89	57.68	54.95	64.89	54.61
LION [71]	65.10	60.15	56.72	54.28	60.61	54.94
Triplane-based method						
NFD [56]	57.55	53.47	54.87	54.06	69.49	71.96
Dense voxel-based method						
NWD [22]	59.78	53.84	56.35	57.98	61.75	58.54
Ours	**52.85**	**49.75**	**53.99**	**48.60**	**57.96**	**54.43**

Table 1. **1-NNA Comparison on ShapeNet [5].** The lower the better. Best scores highlighted in bold.

4.1. Object-level 3D Generation on ShapeNet

Dataset. To benchmark XCube against prior methods, we use the widely-adopted ShapeNet [5] dataset. Following the experimental setup in [35, 56, 70, 71], we choose three specific categories: *Airplane*, *Car* and *Chair*, containing 4145, 7496, 6778 shapes respectively. To build the ground-truth voxel hierarchy for training, we voxelize each mesh at a 512^3 resolution and use the train/val/test split from [10].

Evaluation. To evaluate the geometric quality of our synthesized output, we follow previous work [22, 70, 71] and use *1-NNA* as our main metric (with both Chamfer distance (CD) and earth mover's distance (EMD)). 1-NNA provides a comprehensive measure of both quality and diversity by measuring the distributional similarity between the generated shapes and the validation set [70, 71]. Please refer to the supplementary for more details and evaluations.

Baselines. We compare XCube to state-of-the-art 3D generative models that leverage various latent and shape representations: *PVD* [74], *LION* [71], *NFD [56]*, and *NWD* [22]. PVD and LION employ point clouds as both latent and output-shape representations. NFD uses a triplane-based

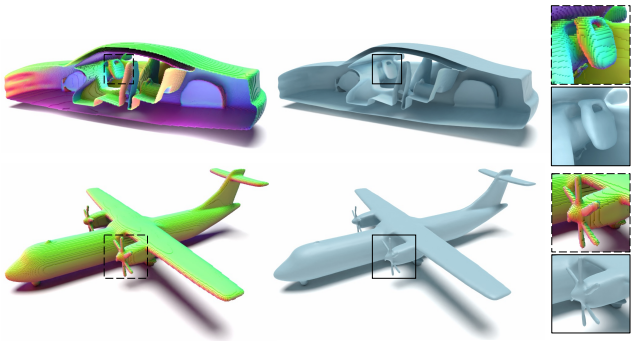


Figure 5. **Close-up View of Our Generated Shape.** The voxel grid is colored by predicted normal. XCube is able to generate a high level of detail, such as the car interior and airplane propellers.

latent representation and decodes a mesh as its output shape representation. NWD uses a dense voxel grid for its latent representation and also decodes a mesh as output. In contrast, our method uses a sparse voxel hierarchy as a latent representation and outputs a sparse voxel grid, which can be readily converted to a mesh.

Results. Tab. 1 provides a quantitative comparison of XCube against baseline approaches and shows the superiority of our approach over past work. The point-based methods are naturally restricted to generating coarse shapes (*i.e.*, 2048 points), while our method is able to generate millions of voxels ($500\times$ larger). The triplane-based method (NFD) exhibits decent performance in categories such as Airplane and Chair with simple geometry. However, its effectiveness diminishes when confronted with the Car category with intricate geometry (such as the suspension of cars), underscoring the challenges of using triplane-based latent representations to generate complex geometric structures. While NWD is based on a dense voxel grid, it is unable to generate fine details due to limited resolution and the fact that its inverse wavelet transform is lossy. In contrast, our method is able to generate high-resolution shapes with fine details, as shown in Fig. 5. Fig. 4 shows a qualitative comparison, which is consistent with our quantitative results.

User-guided Editing. Our method is based on a sparse voxel hierarchy, which is a natural representation for user-guided editing. We demonstrate this ability by allowing users to edit the coarse-level shapes by adding or removing voxels in Goxel [9], a Minecraft-style 3D voxel editor. Fig. 6 shows several examples of user-guided editing.

4.2. Object-level 3D Generation on Objaverse

Dataset. To further demonstrate XCube's ability to perform object-level 3D generation, we evaluate it using the Objaverse [12] dataset, which offers approximately 800K publicly available 3D models. For the text-to-3D experiment, we use the descriptive text automatically generated by Cap3D [36]. For category-conditional generation, we adopt

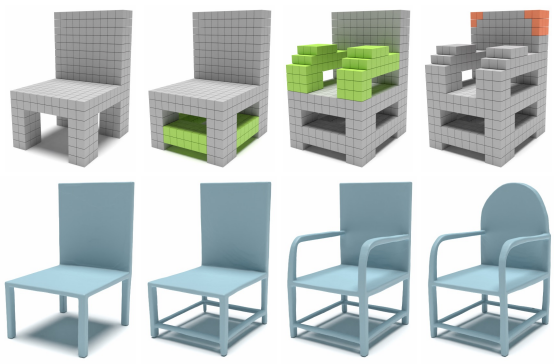


Figure 6. **User-guided Editing.** By adding (**green**) and deleting (**red**) coarse-level voxels, one can easily control the finer 3D shape.

the LVIS categorized object subset of [12], containing $\sim$40K 3D objects. We voxelize the 3D objects at a 512^3 resolution to build the ground truth voxel hierarchy for training.

Evaluation. We conduct a user study on Amazon Mturk to compare our method with Shap·E [24], a state-of-the-art text-to-3D method. Specifically, we gather 30 prompts from popular text-to-3D works [24, 43, 44, 63]. For each prompt, we ask 30 users to pick the 3D object (ours v.s. Shap·E w/o texture) that better matches the text prompt and exhibits higher geometric fidelity. In total, we collect 900 pairwise comparisons. In 79.2% of the comparisons, participants vote for our generated 3D objects.

Results. We provide qualitative results for text-to-3D in Fig. 7, and category-conditional generation in Fig. 9. The texture of our results is generated by off-the-shelf texture synthesis methods [4, 51]. We also compare our method with Shap·E [24] in Fig. 8. Our whole pipeline including generating the geometry and the texture for one object takes about 1 minute (30s for objects and 30s for textures). We observe that our method is able to generate more diverse 3D objects with higher geometric fidelity and finer details than Shap·E, as shown in Fig. 9.

4.3. Large-scale Scene-level 3D Generation

Dataset. To demonstrate our model's scalability and ability to generate large-scale high-resolution scenes, we train and evaluate it on the Waymo Open Dataset [60] which contains 1000 LiDAR scan sequences capturing different driving scenarios. Here, we extract the ground-truth dense scene geometry by accumulating the LiDAR scans and propagating the semantic labels. To construct the ground-truth sparse voxel hierarchy, we crop each of the extracted scenes to $102.4\,\mathrm{m}\times\ 102.4\,\mathrm{m}$ chunks and voxelize the points and meshes at a resolution of 1024^3, resulting in a voxel size of $10\,\mathrm{cm}$. To demonstrate the superiority of our representation power, we train a model on Karton City [1], a custom synthetic dataset of 20 blocks of synthetic city scenes using the same resolution settings as Waymo. We divide the 20 blocks

Figure 7. **Text-to-3D Results on Objaverse [12].** For each sample we show the input text prompt, the generated sparse voxel grid colored by normal, the extracted mesh, and the textured mesh (using [51]) sequentially.

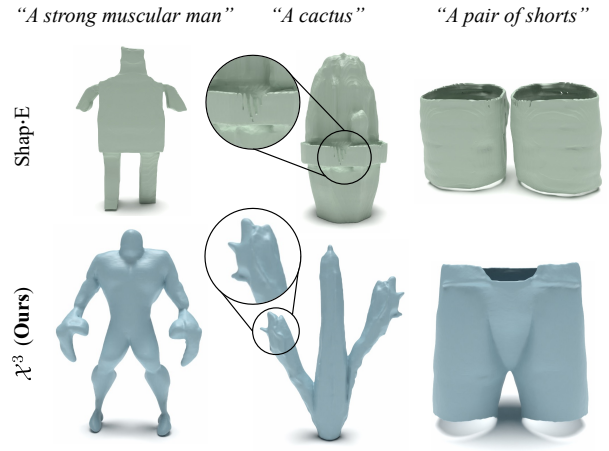


Figure 8. **Comparison with Shap·E [24].** We can generate high-quality shapes that better match the given prompts.

into train/val splits and randomly crop $102.4\,\mathrm{m} \times 102.4\,\mathrm{m}$ chunks in each split to generate 900/100 unique train/val scenes. Fig. 12 shows examples from the Karton City model and we include more results in the supplementary.

Evaluation. To evaluate the quality of our generated results, we perform a user study using Amazon Mturk. Here we show 30 users 30 pairs of scenes (totaling 900 pairwise comparisons) where one scene is sampled from the validation set and the other is a random output generated by our model. We ask each user to rank which scene is more realistic. Out of the 900 comparisons, 66.3% favors our results over the ground truth, demonstrating that our generated outputs are of high quality. Fig. 10 shows several unconditional generations from our model as well as decoded semantics and normals.

Single-Scan Conditioning. We demonstrate that our model can be used to perform conditional generation on large-scale scenes. In this qualitative experiment, we condition the model on a single input LiDAR scan and generate a complete

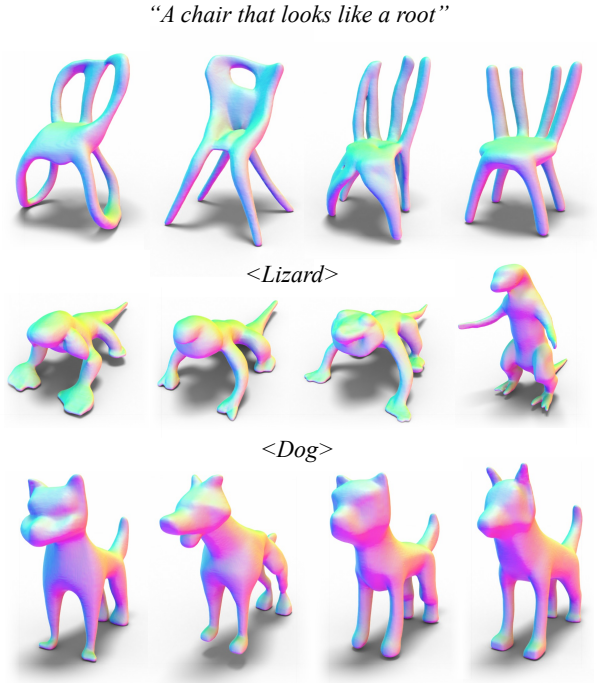


Figure 9. **Diversity of Our Generated Shapes.** XCube can generate diverse shapes under the same text prompt or category label.

scene with normals and semantics. Fig. 11 shows several completions using our method. Note that our input does not contain semantics, yet our model is able to generate plausible geometric and semantic completion results. The supplementary shows additional details and figures.

4.4. Ablation Study

Progressive Pruning. We replace the progressive pruning part of our pipeline with a single pruning step for the $16^3 \rightarrow 128^3$ VAE on ShapeNet Chairs. We observe that the reconstruction accuracy (grid IoU) drops from 92.88% to 89.68%, indicating that progressive pruning is critical for

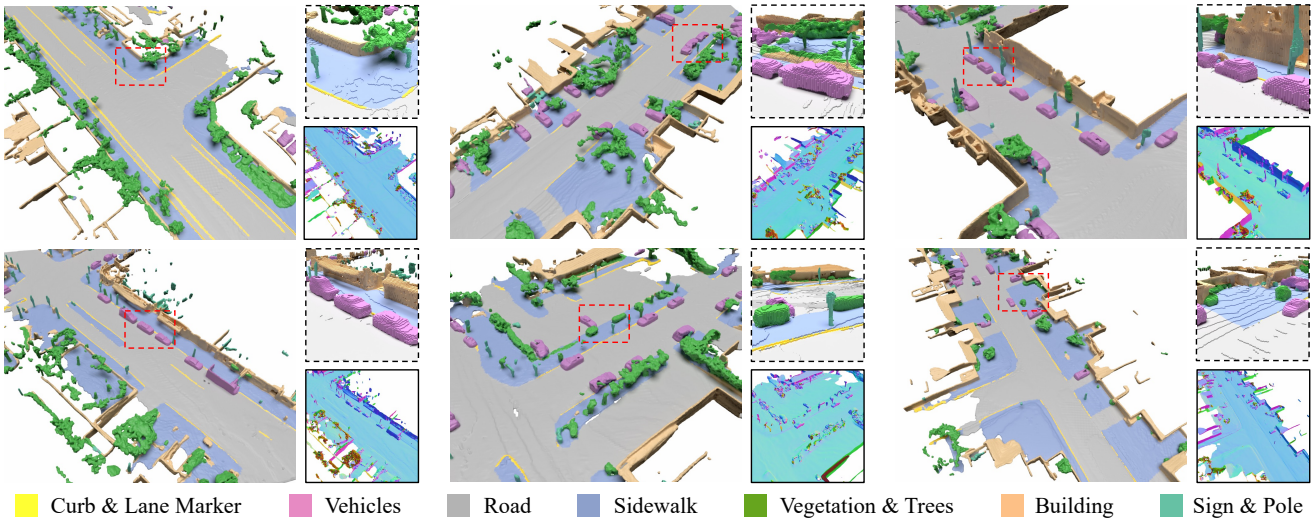


Figure 10. **Unconditional Samples on the Waymo Open Dataset [60].** The dashed boxes show a zoomed-in view and the solid boxes show the normal map for the extracted mesh. Best viewed with 200% zoom-in.

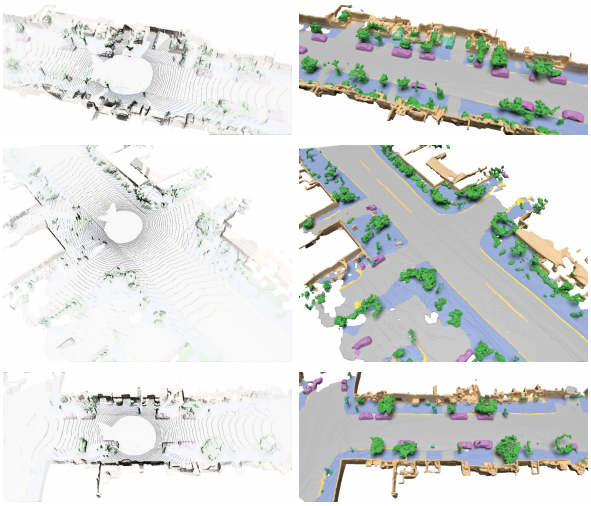


Figure 11. **Single-scan-conditioned Generation.** The left column shows the input LiDAR scan and the right column shows our generated semantic mesh conditioned on the input.

preserving shape details and injecting 3D inductive bias. Furthermore, the GPU memory usage also increases by a factor of $3\times$ when removing progressive pruning.

Hierarchy Configuration. As shown in Tab. 2, we compare the performance of our model with different hierarchy configurations on ShapeNet Chairs. We observe that: (1) the hierarchical model outperforms the single-level model, emphasizing the importance of a sparse voxel hierarchy in 3D generative modeling. (2) the model's performance is robust to the resolution of the initial hierarchy level. We find 16^3 is sufficient for capturing the overall shape of the object. (3) using two-level and three-level models achieve comparable performance. For unconditional generation, we use a two-level model for fast sampling. And for the user-editing

Model	CD (%)	EMD (%)
$16^3 \rightarrow 512^3$	59.31	57.46
$16^3 \rightarrow 128^3 \rightarrow 512^3$	53.99	**48.60**
$32^3 \rightarrow 128^3 \rightarrow 512^3$	55.39	51.40
$4^3 \rightarrow 16^3 \rightarrow 128^3 \rightarrow 512^3$	**52.88**	53.62

Table 2. **Ablation of Different Resolutions and Depths of the Hierarchy.** Metrics are in 1-NNA. The lower the better.

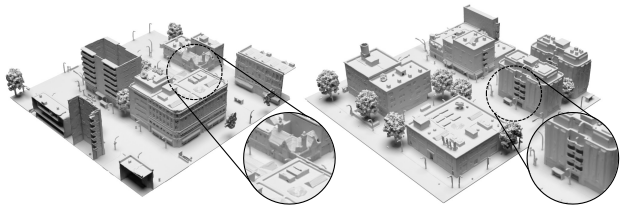


Figure 12. **Unconditional Samples on Karton City [1].**

setting, we use a three-level model for easier editing.

5. Discussion

Conclusion. We presented $\mathcal{X}^3$, a novel generative model for large-scale 3D scenes represented as a hierarchy of sparse 3D voxel grids. We proposed a hierarchical voxel latent diffusion model that learns the joint distribution of the latent representation and the sparse voxel hierarchy. The effectiveness of our method was demonstrated on both object-level and scene-level generation, reflecting our method's capability of generating high-resolution 3D scenes with fine details.

Limitations and Future Work. Nevertheless, there are challenges that remain. Due to current 3D datasets being still not on par with image datasets (such as LAION-5B [54]), our text-to-3D model is hard to deal with complex prompts. In the future, we plan to extend our method in the setting of image-conditioning, as well as leveraging the learned prior as a fundamental model to support more downstream tasks.

References

[1] 3d karton city model. https://www.turbosquid.com/3d-models/3d-karton-city-2-model-1196110, 2023. Accessed: 2023-08-01. 2, 5, 6, 8

[2] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning representations and generative models for 3d point clouds. In *International Conference on Machine Learning (ICML)*, pages 40–49, 2018. 2

[3] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22563–22575, 2023. 2, 4

[4] 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 International Conference on Computer Vision (ICCV)*, 2023. 6

[5] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1, 2, 5

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, pages 333–350, 2022. 2

[7] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5939–5948, 2019. 2

[8] Zhaoxi Chen, Guangcong Wang, and Ziwei Liu. Scenedreamer: Unbounded 3d scene generation from 2d image collections. *arXiv preprint arXiv:2302.01330*, 2023. 2

[9] Guillaume Chereau. Goxel: 3d voxel editor, 2023. Accessed: 2023-11-16. 6

[10] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European Conference on Computer Vision (ECCV)*, pages 628–644, 2016. 5

[11] 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. *arXiv preprint arXiv:2307.05663*, 2023. 2

[12] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13142–13153, 2023. 1, 2, 5, 6, 7

[13] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, pages 8780–8794, 2021. 4, 14

[14] William Falcon and The PyTorch Lightning team. PyTorch Lightning, 2019. 14

[15] 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. In *Advances in Neural Information Processing Systems*, 2022. 2

[16] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. SDM-NET: deep generative network for structured deformable mesh. *ACM Transactions on Graphics (TOG)*, 38(6):243:1–243:15, 2019. 2

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 2

[18] 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. 1, 3

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020. 2, 4

[20] Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 2022. 4, 5

[21] Jiahui Huang, Zan Gojcic, Matan Atzmon, Or Litany, Sanja Fidler, and Francis Williams. Neural kernel surface reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4369–4379, 2023. 3, 5, 12

[22] Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3d shape generation. In *SIGGRAPH Asia*, 2022. 1, 2, 5

[23] Moritz Ibing, Gregor Kobsik, and Leif Kobbelt. Octree transformer: Autoregressive 3d shape generation on hierarchically structured sequences. *arXiv preprint arXiv:2111.12480*, 2021. 2

[24] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. 2, 6, 7

[25] Seung Wook Kim, Bradley Brown, Kangxue Yin, Karsten Kreis, Katja Schwarz, Daiqing Li, Robin Rombach, Antonio Torralba, and Sanja Fidler. Neuralfield-ldm: Scene generation with hierarchical latent diffusion models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8496–8506, 2023. 2

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 14

[27] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 2

[28] Muheng Li, Yueqi Duan, Jie Zhou, and Jiwen Lu. Diffusion-sdf: Text-to-shape via voxelized diffusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12642–12651, 2023. 2

[29] 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 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 300–309, 2023. 2

[30] Chieh Hubert Lin, Hsin-Ying Lee, Willi Menapace, Menglei Chai, Aliaksandr Siarohin, Ming-Hsuan Yang, and Sergey Tulyakov. Infinicity: Infinite-scale city synthesis. *arXiv preprint arXiv:2301.09637*, 2023. 2

[31] Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. *arXiv preprint arXiv:2311.07885*, 2023. 1

[32] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Zexiang Xu, Hao Su, et al. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *arXiv preprint arXiv:2306.16928*, 2023. 2

[33] 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. *arXiv preprint arXiv:2303.11328*, 2023. 1, 2

[34] Andrew Luo, Tianqin Li, Wen-Hao Zhang, and Tai Sing Lee. Surfgen: Adversarial 3d shape synthesis with explicit surface discriminators. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 16218–16228, 2021. 2

[35] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2837–2845, 2021. 2, 5

[36] Tiange Luo, Chris Rockwell, Honglak Lee, and Justin Johnson. Scalable 3d captioning with pretrained models. In *Advances in Neural Information Processing Systems*, 2023. 6

[37] 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. In *ECCV*, 2020. 13

[38] Ken Museth. VDB: high-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics (TOG)*, 32(3):27:1–27:22, 2013. 2, 5, 12

[39] Ken Museth. Nanovdb: A gpu-friendly and portable vdb data structure for real-time rendering and simulation. In *ACM SIGGRAPH 2021 Talks*, New York, NY, USA, 2021. Association for Computing Machinery. 12

[40] George Kiyohiro Nakayama, Mikaela Angelina Uy, Jiahui Huang, Shi-Min Hu, Ke Li, and Leonidas Guibas. Difffacto: Controllable part-based 3d point cloud generation with cross diffusion. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2023. 2

[41] Gimin Nam, Mariem Khlifi, Andrew Rodriguez, Alberto Tono, Linqi Zhou, and Paul Guerrero. 3d-ldm: Neural implicit 3d shape generation with latent diffusion models. *arXiv preprint arXiv:2212.00842*, 2022. 2, 3

[42] Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *International Conference on Machine Learning (ICML)*, pages 7220–7229, 2020. 2

[43] 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. 2, 6

[44] Evangelos Ntavelis, Aliaksandr Siarohin, Kyle Olszewski, Chaoyang Wang, Luc Van Gool, and Sergey Tulyakov. Autodecoding latent 3d diffusion models. In *Advances in Neural Information Processing Systems*, 2023. 2, 6

[45] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, pages 523–540, 2020. 12, 13

[46] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023. 2, 13

[47] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023. 1

[48] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763, 2021. 4

[49] Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, Alejandro Newell, Hei Law, Ankit Goyal, Kaiyu Yang, and Jia Deng. Infinite photorealistic worlds using procedural generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12630–12641, 2023. 2

[50] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*, pages 1530–1538, 2015. 2

[51] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. In *SIGGRAPH*, 2023. 6, 7, 13, 14

[52] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, 2022. 2, 3, 4

[53] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022. 4

[54] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5B: an open large-scale dataset for training next generation image-text models. In *Advances in Neural Information Processing Systems*, 2022. 8

[55] Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-e: Text-guided voxel editing of 3d objects. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2023. 2

[56] J. Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20875–20886, 2023. 2, 5

[57] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*, pages 2256–2265, 2015. 2

[58] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 4

[59] Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould. 3d-gpt: Procedural 3d modeling with large language models. *arXiv preprint arXiv:2310.12945*, 2023. 2

[60] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2443–2451, 2020. 2, 5, 6, 8

[61] Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Viewset diffusion:(0-) image-conditioned 3d generative models from 2d data. *arXiv preprint arXiv:2306.07881*, 2023. 2

[62] Haotian Tang, Zhijian Liu, Xiuyu Li, Yujun Lin, and Song Han. Torchsparse: Efficient point cloud inference engine. *MLSys*, 2022. 2, 5, 12

[63] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023. 6

[64] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2107–2115, 2017. 2

[65] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. In *Advances in Neural Information Processing Systems*, pages 11287–11302, 2021. 2, 3

[66] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016. 2

[67] 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. *arXiv preprint arXiv:2305.16213*, 2023. 1, 2

[68] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, pages 82–90, 2016. 2

[69] Haozhe Xie, Zhaoxi Chen, Fangzhou Hong, and Ziwei Liu. Citydreamer: Compositional generative model of unbounded 3d cities. *arXiv preprint arXiv:2309.00610*, 2023. 2

[70] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge J. Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4540–4549, 2019. 5

[71] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. LION: latent point diffusion models for 3d shape generation. In *Advances in Neural Information Processing Systems*, 2022. 2, 3, 5, 14

[72] Dongsu Zhang, Changwoon Choi, Jeonghwan Kim, and Young Min Kim. Learning to generate 3d shapes with generative cellular automata. In *ICLR*, 2021. 2

[73] Xin-Yang Zheng, Hao Pan, Peng-Shuai Wang, Xin Tong, Yang Liu, and Heung-Yeung Shum. Locally attentional SDF diffusion for controllable 3d shape generation. *ACM Transactions on Graphics (TOG)*, 42(4):91:1–91:13, 2023. 2

[74] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5806–5815, 2021. 5

Appendices

In this supplementary material, we provide additional details on our method and experiments. In Appendix A, we describe our sparse 3D deep learning framework, and compare it to state-of-the-art implementations. In Appendix B, we provide more implementation details for our method as well as precise definitions of our loss function and evaluation metrics. In Appendix C, we provide more qualitative results on all the datasets we trained/evaluated on in the main paper. We additionally provide a **supplementary video** in the accompanying files to better illustrate our results.

A. Sparse 3D Learning Framework

All of our networks are implemented using a customized sparse 3D deep learning framework built on top of PyTorch. To represent sparse grids of features and perform efficient deep-learning operations (convolution, pooling, etc.) over them, we leverage NanoVDB [39], a GPU-friendly implementation of the VDB data structure [38]. A VDB tree is a variant of B+-Tree with four layers where the top layer is a hash table, followed by two internal layers (with branching factor $32^3, 16^3$), followed by leaf nodes with 8^3 voxels.

To demonstrate the effectiveness of our VDB-based deep learning framework, we benchmark it against TorchSparse [62], a state-of-the-art sparse deep learning framework. As shown in Tab. 3, our custom framework is both fast and memory-efficient, especially for large input grid resolutions. Built upon the highly efficient VDB data structure, our 3D representation is compactly stored in memory and supports more efficient nearest neighbor lookup and processing than its hash table counterpart in [62]. Such a framework lays the foundation for our high-resolution 3D generative model and has potential to applications in many other downstream tasks such as reconstruction and perception.

	Voxel Grid Memory (MB) ↓		Convolution Forward Time (ms) ↓		
Grid Resolution	512^3	1024^3	32^3	256^3	1024^3
TorchSparse [62]	15.0	104.6	2.1	8.5	446.0
Ours	**3.6**	**8.4**	**0.5**	**5.0**	**149.6**

Table 3. Sparse 3D benchmark results.

B. Implementation Details

B.1. Loss Definition

Our model is able to output various attributes $\mathbf{A}$ defined on the voxel grids. Here we omit the level subscript l for simplicity. The direct output of the network at each voxel at each level includes surface normal $\boldsymbol{n} \in \mathbb{R}^3$, semantic label $\boldsymbol{s} \in \mathbb{R}^S$, and neural kernel features $\boldsymbol{\phi} \in \mathbb{R}^4$. Here the neural kernel features $\boldsymbol{\phi}$ are used for computing continuous TSDF values in 3D space for highly-detailed *subvoxel*-level surface extraction (using the techniques from [21]), and it could also be replaced with implicit features $\boldsymbol{q}$ to extract TUDF values for open surfaces as in [45]. The attribute loss $\mathcal{L}^{\text{Attr}}$, as mentioned in Eq. (6) of the main text, is a mixture of different supervisions, written as follows:

$$\mathcal{L}^{\text{Attr}} = \lambda_1 \underbrace{\|\boldsymbol{n} - \boldsymbol{n}_{\text{GT}}\|_2^2}_{\text{normal loss}} + \lambda_2 \underbrace{\text{BCE}(\boldsymbol{s}, \boldsymbol{s}_{\text{GT}})}_{\text{semantic loss}} + \lambda_3 \underbrace{\mathbb{E}_{\boldsymbol{x}\in\mathbb{R}^3} \|f(\boldsymbol{x}) - \text{TSDF}(\boldsymbol{x}, \boldsymbol{X}_{\text{GT}})\|_1}_{\text{surface loss}}, \tag{8}$$

where $\boldsymbol{n}_{\text{GT}}$ and $\boldsymbol{s}_{\text{GT}}$ are the ground-truth normal and semantic label at each voxel, and $\boldsymbol{X}_{\text{GT}}$ is the ground-truth dense point cloud of the surface. The surface loss is computed by sampling points $\boldsymbol{x}$ in the 3D space and comparing the predicted TSDF values $f(\boldsymbol{x})$ with the ground-truth TSDF values $\text{TSDF}(\boldsymbol{x}, \boldsymbol{X}_{\text{GT}})$. To compute $f(\boldsymbol{x})$ given arbitrary input positions, we leverage the predicted neural kernels $\boldsymbol{\phi}$ to solve for a surface fitting problem as in [21]:

$$f(\boldsymbol{x}) = \sum_v \alpha_v K(\boldsymbol{x}, \boldsymbol{x}_v) = \sum_v \alpha_v \boldsymbol{\phi}_v^\top \boldsymbol{\phi}(\boldsymbol{x}) K_b(\boldsymbol{x}, \boldsymbol{x}_v), \tag{9}$$

where v is the index of the voxels, $\boldsymbol{\phi}(\boldsymbol{x})$ is the neural kernel evaluated at the input position $\boldsymbol{x}$ using bezier interpolation from its nearby voxels, and $K_b(\boldsymbol{x}, \boldsymbol{x}_v) = B(\boldsymbol{x} - \boldsymbol{x}_v)$ is a shift-invariant Bezier kernel. The coefficients α_v are obtained by performing a linear solve as detailed in [21]. Similarly, for open surfaces we can replace the neural kernels $\boldsymbol{\phi}$ with implicit features $\boldsymbol{q}$

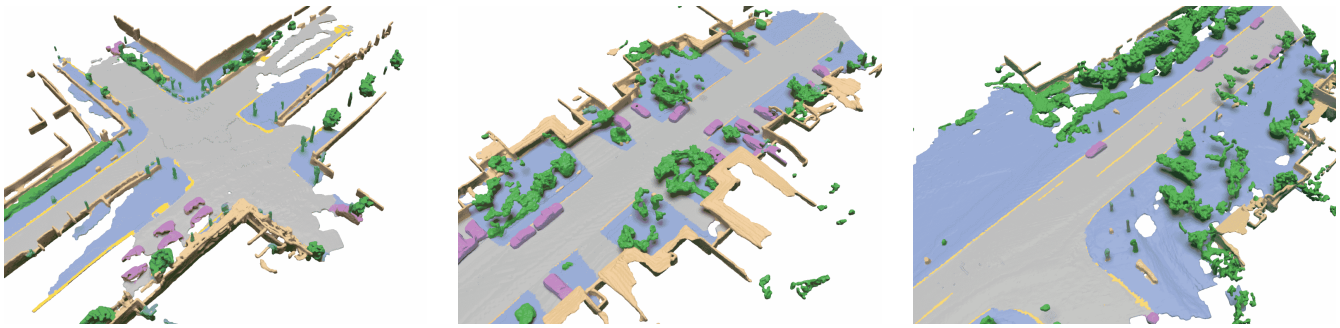


Figure 13. Results of micro-conditioning on Waymo dataset. The voxel number conditioning increases from left to right. There is a clear trend of increasing number of voxels and more diverse contents in the sampled scenes.

and define $f(\boldsymbol{x})$ as a local MLP function digesting trilinearly interpolated $\boldsymbol{q}$ at position $\boldsymbol{x}$ [45]. We set $\lambda_1 = 1$, $\lambda_2 = 15$, and $\lambda_3 = 1$ in our experiment. For the KL divergence, we normalize it by the number of voxels of the voxel grid and then use a loss weight $\lambda = 0.0015$ for all our experiments.

B.2. Conditioning

We explore diverse condition settings for our voxel diffusion models: (1) For the associated attributes from the previous level, we optionally concatenate them to the latent feature $\mathbf{X}$ before the latent diffusion. For example, for user-control cases, we do not concatenate them for flexibly adding or deleting voxels. (2) For the text prompts, we use cross-attention to fuse them into the latent. (3) For the category condition, we use AdaGN and fuse them with timestep embedding by adding. (4) For single scan conditions, we use an additional point encoder to quantize the single scan point cloud to a voxel grid and concatenate it with the latent feature $\mathbf{X}$.

Micro-conditioning. We found that the Waymo dataset suffers from missing voxels due to the sparsity of the LiDAR scans. To mitigate this issue, we use a micro-conditioning scheme following SD-XL [46] to inject additional condition to the diffusion backbone describing the number of the voxels. This helps when the dataset itself contains multi-modal distributions, and allows fine-grained control of the generated scale of the scene.

B.3. Texture Synthesis

While our model is focused on generating 3D geometry, we also explore the possibility of generating textures for the generated shapes. To this end, we use a state-of-the-art texture generator TEXTure [51] to create texture maps for the generated shapes. The method works by applying a sequence of depth-conditioned stable diffusion models to multiple views of the shape. Later steps in the process are conditioned on the previous steps, allowing the model to generate consistent textures with smooth transitions. We choose to decouple the geometry and texture generation processes to allow for more flexibility and controlability – *e.g.*, given the same geometry, different textures can be generated and selected. We demonstrate the effectiveness of the full pipeline on Objaverse dataset and use the same text prompts for both the geometry and the texture. Results are shown in Fig. 14.

B.4. Network Architecture

Variational Autoencoders (VAE). We use a custom Autoencoder architecture for our VAE. Given an input voxel grid, $\mathbf{G}_l$ at level l, and associated per-voxel attributes $\mathbf{A}_l$, we first positionally encode each voxel using the same function as [37] and then concatenate the positional encoding of each voxel with the corresponding attribute. We then apply a linear layer to the concatenated positional embedding and attribute to lift it to a d-dimensional feature (Where d is chosen depending on the task and described in Table 4). Our VAE then applies successive convolution and max pooling layers, coarsening the voxels to a bottleneck dimension. When $l = 1$ (*i.e.* the coarsest level of the hierarchy), we zero pad the bottleneck layer into a dense tensor, otherwise, the bottleneck is a sparse tensor. We then apply 4 convolutional layers to convert the bottleneck tensor into a latent tensor $\mathbf{X}$ of the same shape and sparsity pattern as the bottleneck. Latent diffusion is done over the tensor $\mathbf{X}$. At the end of the decoder, we apply attributes-specific heads (MLPs) to predict the associated attributes within each voxel. Hyperparameters for our VAEs are listed in Tab. 4.

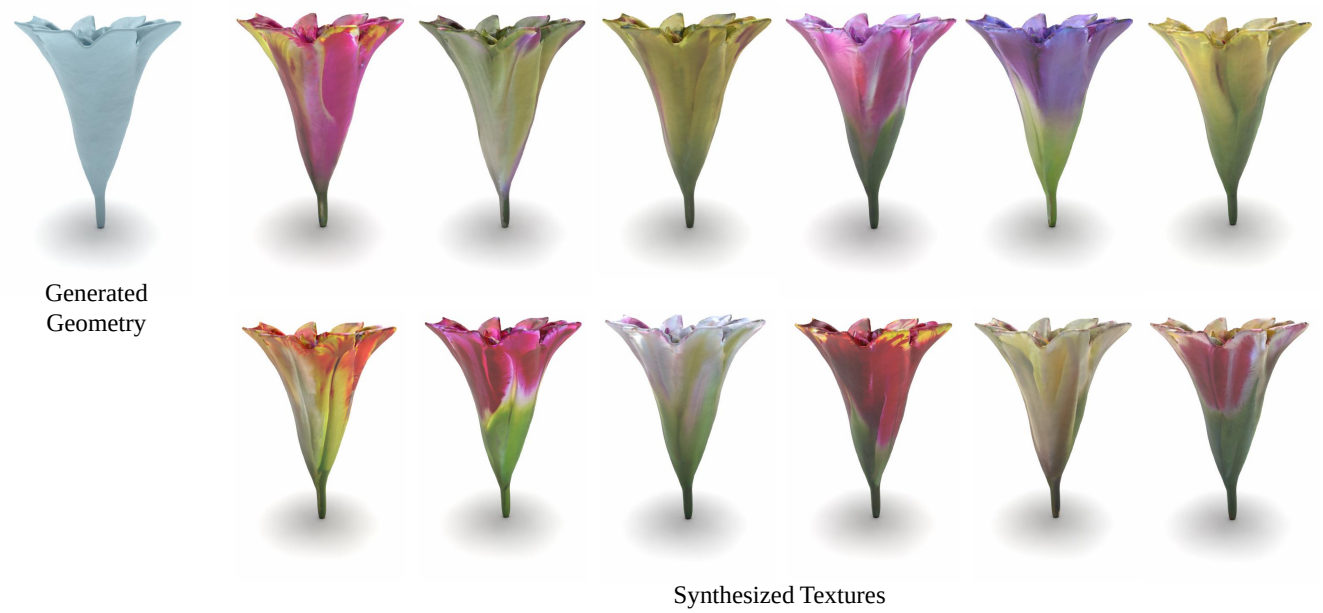


Figure 14. Diverse texture synthesis results. Based on the same generated geomtry, we could generate diverse textures by using TEXTrue [51].

Diffusion UNet. As mention in the main paper, we adopt a a 3D sparse variant of the backbone used in [13] for our voxel latent diffusion. Hyperparameters for training them are in Tab. 5

B.5. Training Details

We train all of our models using Adam [26] with $\beta_1 = 0.9$ and $\beta_1 = 0.999$. We use an EMA rate of 0.9999 for all experiments and use PyTorch Lightning [14] for training. For ShapeNet models, we use $8\times$ NVIDIA Tesla V100s for training. For other datasets, we use $8\times$ NVIDIA Tesla A100s for training.

B.6. Metric Definition

To perform a quantitative comparison of our generative model on the ShapeNet dataset, we leverage the framework used in [71] which uses the *1-NNA* metric defined as follows: Given a generated set of point clouds S_g, a reference set of point clouds S_r, and a metric $D(\cdot,\cdot) : 2^{\mathbb{R}^3} \times 2^{\mathbb{R}^3} \to \mathbb{R}$ between two point clouds, the 1-NNA metric is defined as

$$\text{1-NNA}(S_g, S_r) = \frac{\sum_{X \in S_g} \mathbb{1}[N_X \in S_r] + \sum_{Y \in S_r} \mathbb{1}[N_Y \in S_r]}{|S_g| + |S_r|}, \tag{10}$$

where N_A is the closest point cloud to $A \in 2^{\mathbb{R}^3}$ in the set $S_g \cup S_r - \{X\}$ under the metric $D(\cdot,\cdot)$ (*i.e.* the closest point cloud to A in the generated and reference set not including A itself), and $\mathbb{1}[\cdot]$ is the indicator function which returns 1 if the argument is true and 0 otherwise.

Intuitively, the 1-NNA distance is the classification accuracy when using nearest neighbors under D to determine if a point cloud was generated ($\in S_g$) or not ($\in S_r$). If the generated set is close in distribution to the reference set, then the classification accuracy should be around 50% which is the best 1-NNA score achievable.

In our experiment, we sampled 2048 points from the surface of each shape (following [71]) to generate S_g and S_r and used the Chamfer and Earth Mover's distances as metrics D to compute the 1-NNA.

C. More results

In this section, we provide more qualitative results on all datasets. First, we show more text-to-3D results on Objaverse in Fig. 16 to 18. Then, we show more results on ShapeNet in Fig. 15 and 19 to 21. Despite the high quality of our generated shapes, we show that our model does not overfit the training samples and is able to generate novel shapes in Fig. 15 by retrieving the most similar shapes in the training set given the generated samples. Furthermore, we show more results on Waymo in Fig. 22 and 23. Finally, we show more results on Karton City in Fig. 24.

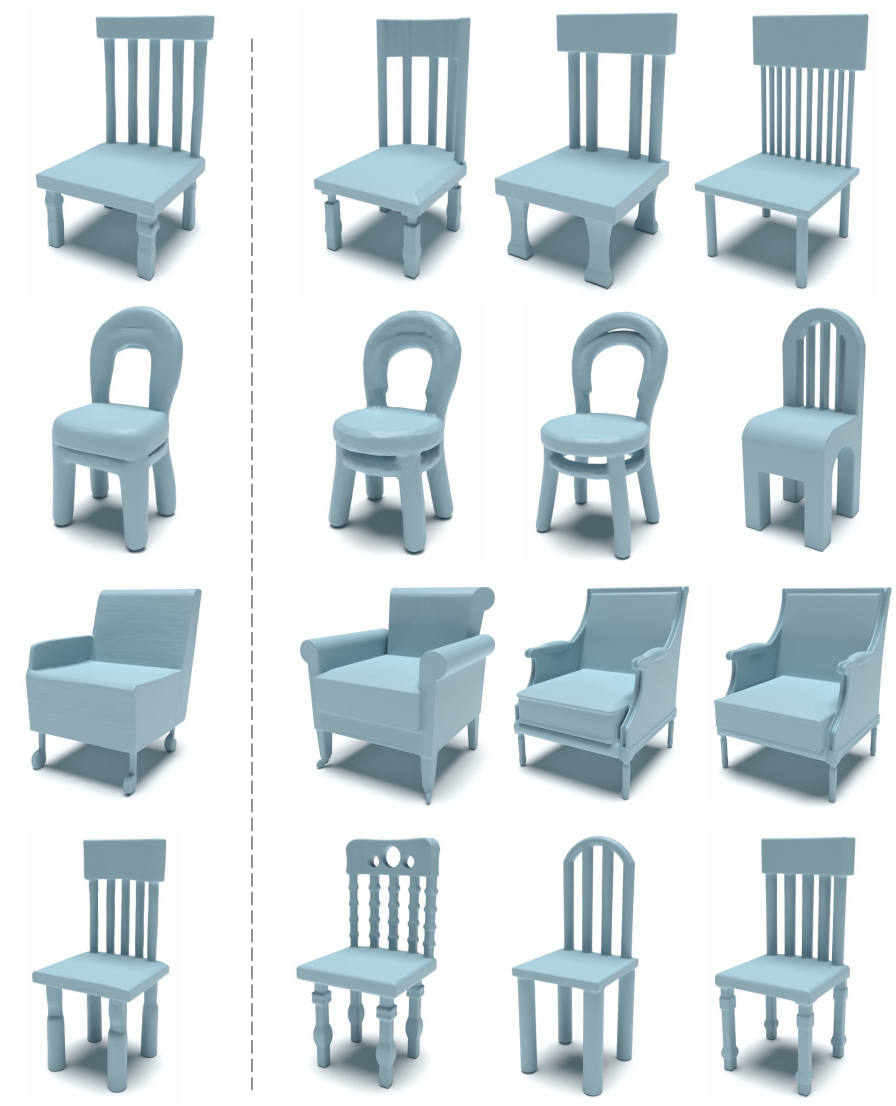


Figure 15. Shape Novelty Analysis. From our generated shape (left), we retrieve top-three most similar shapes in training set by CD distance

"A 3D model of lion"

"A campfire"

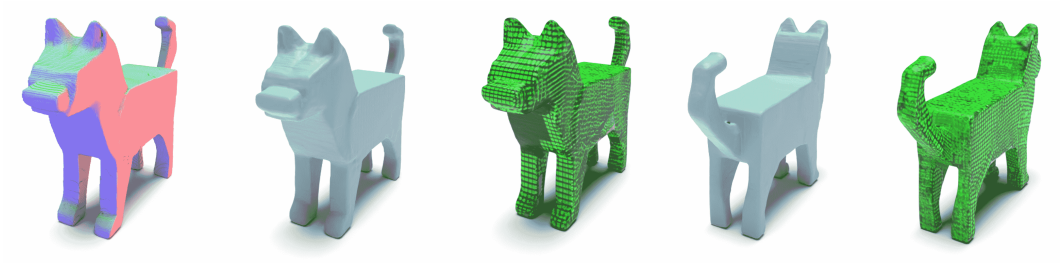
"A 3D model of croissant"

"A 3D model of eagle head"

"A 3D model of dragon head"

Figure 16. More qualitative results on text-to-3D.

"A voxelized dog"



"A diamond ring"

"A 3D model of cat"

"A 3D model of duck"

Figure 17. More qualitative results on text-to-3D.

"A designer dress"

"A 3D model of koala"

"A 3D model of mushroom"

"A fireplug"

Figure 18. More qualitative results on text-to-3D.

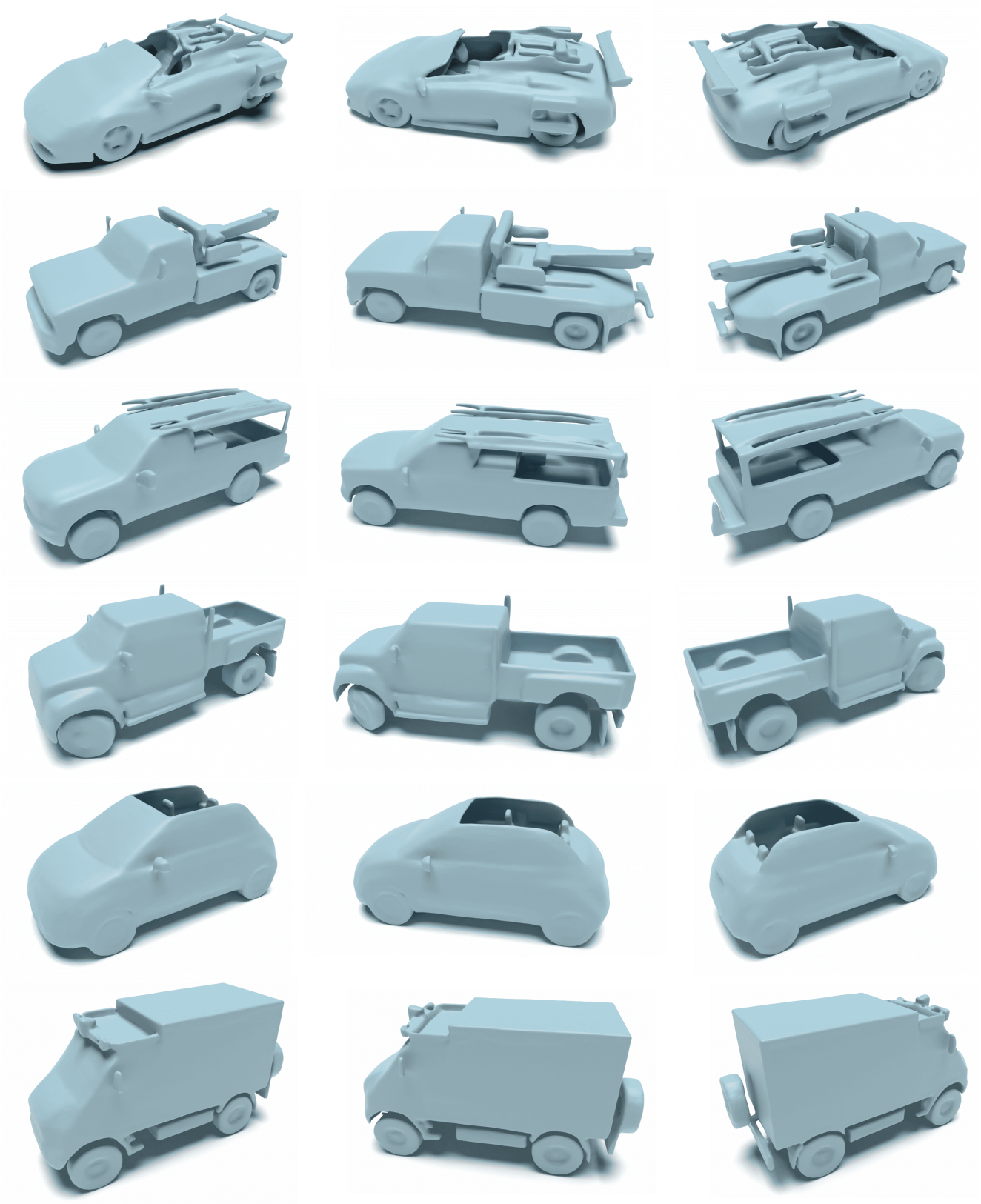


Figure 19. More qualitative results on ShapeNet Car.

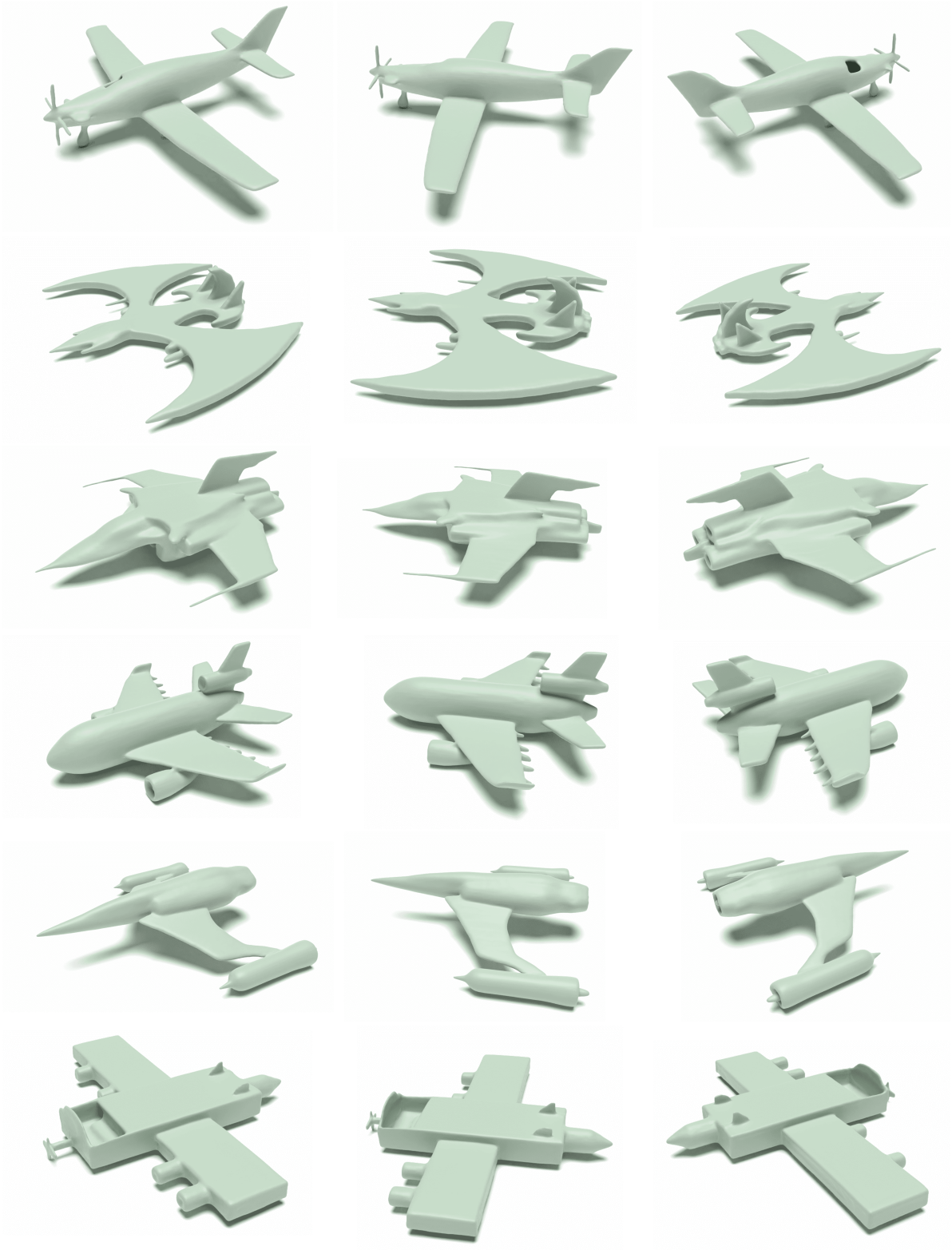


Figure 20. More qualitative results on ShapeNet Airplane.

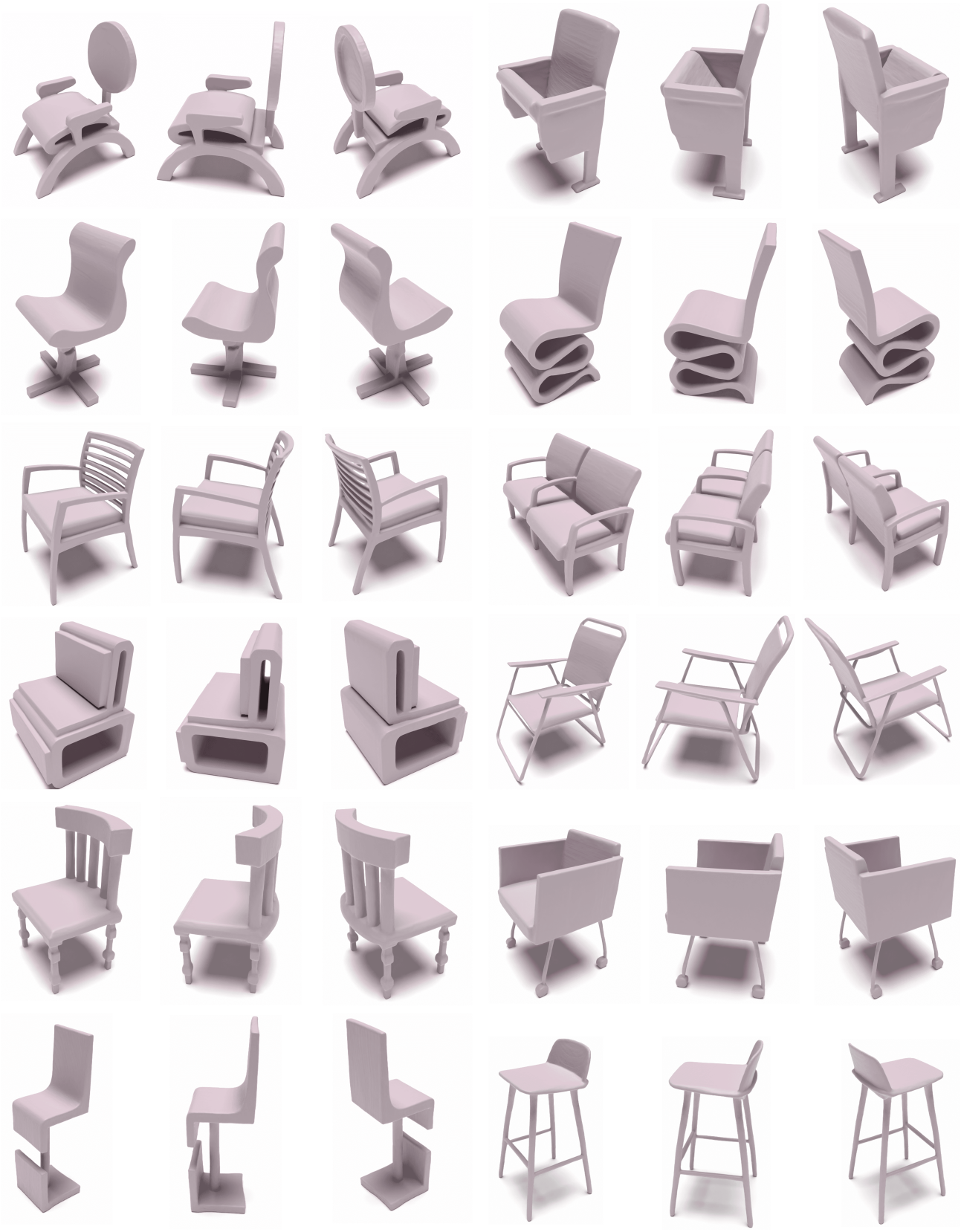


Figure 21. More qualitative results on ShapeNet Chair.

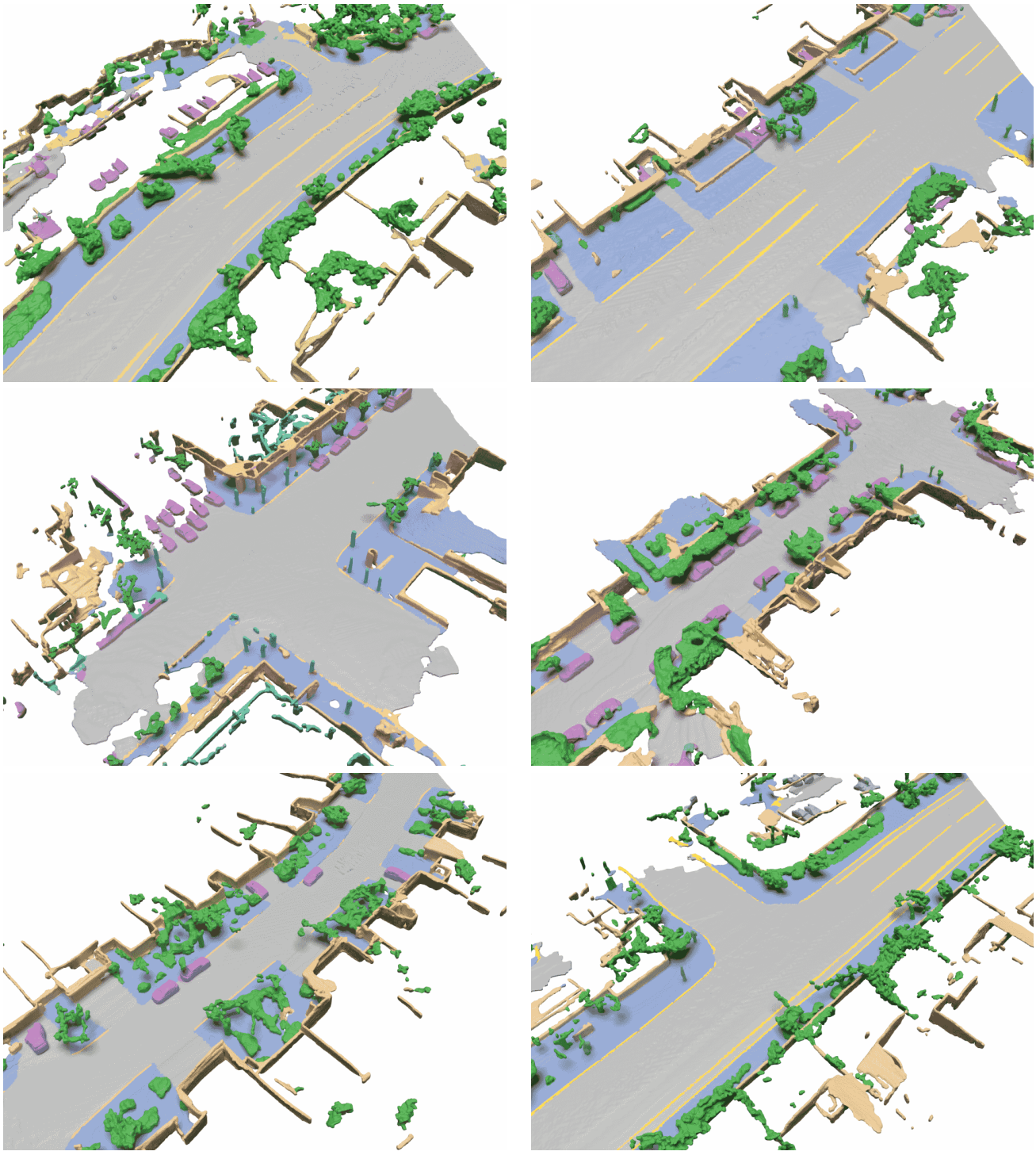


Figure 22. More qualitative results on Waymo.

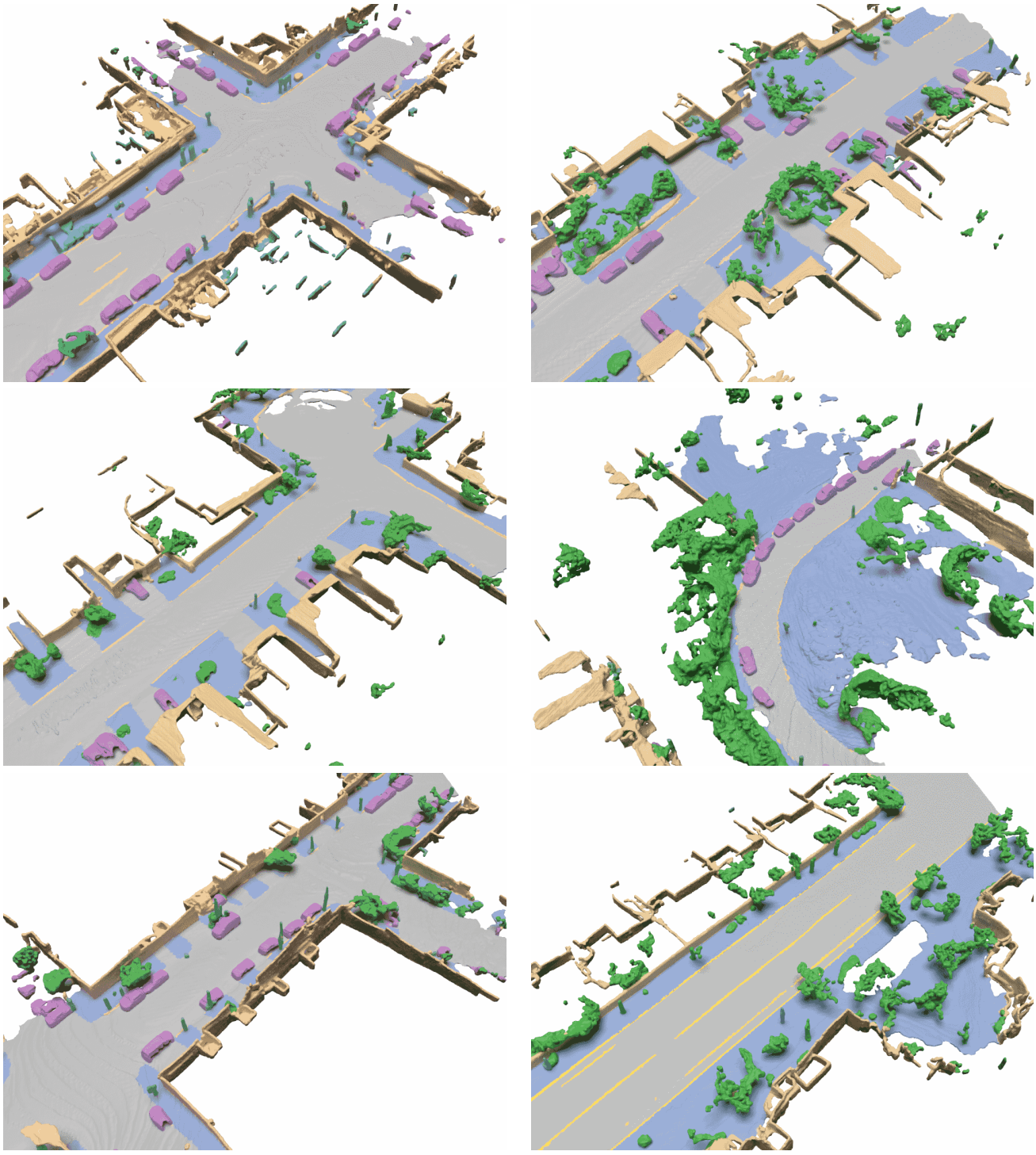


Figure 23. More qualitative results on Waymo.

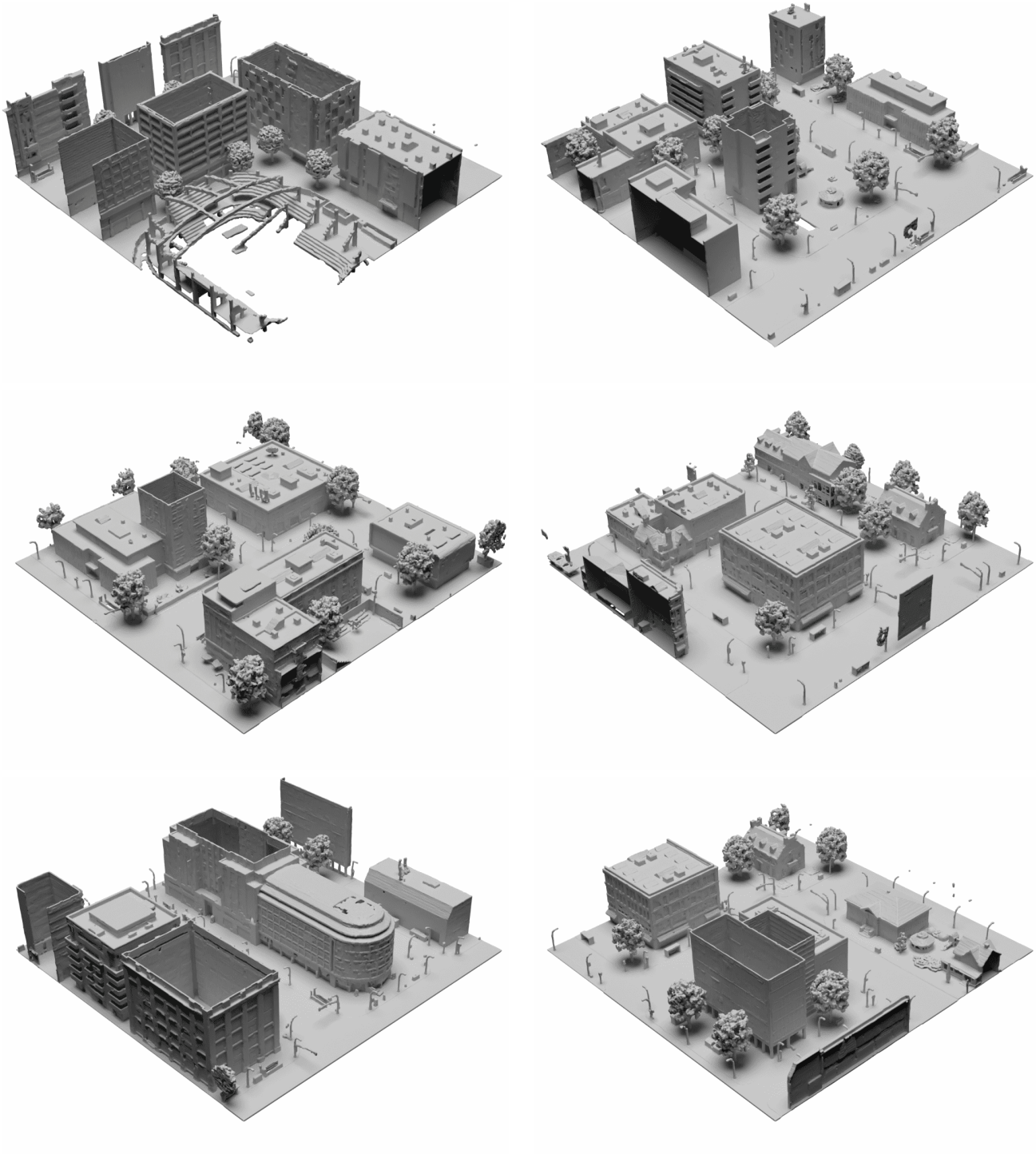


Figure 24. More qualitative results on Karton City.

	ShapeNet $16^3 \rightarrow 128^3$	ShapeNet $128^3 \rightarrow 512^3$	Objaverse $16^3 \rightarrow 128^3$	Objaverse $128^3 \rightarrow 512^3$	Waymo $32^3 \rightarrow 256^3$	Waymo $256^3 \rightarrow 1024^3$
Model Size	59.6M	3.8M	236M	14.9M	59.4M	3.8M
Base Channels	64	32	128	64	64	32
Channels Multiple	1,2,4,8	1,2,4	1,2,4,8	1,2,4	1,2,4,8	1,2,4
Latent Dim	16	8	16	8	16	8
Batch Size	16	32	32	32	32	32
Epochs	100	100	25	10	50	50
Learning Rate	1e-4					

Table 4. **Hyperparameters for VAE.** For the Karton City dataset, we used the same hyperparameters as the Waymo dataset.

	ShapeNet - 16^3	ShapeNet - 128^3	Objaverse - 16^3	Objaverse - 128^3	Waymo - 32^3	Waymo - 256^3
Diffusion Steps	1000					
Noise Schedule	linear					
Model Size	691M	79.6M	1.5B	79.6M	702M	76.6M
Base Channels	192	64	256	64	192	64
Depth	2					
Channels Multiple	1,2,4,4	1,2,2,4	1,2,4,4	1,2,2,4	1,2,4,4	1,2,2,4
Heads	8					
Attention Resolution	16,8,4	32,16	16,8,4	32,16	16,8	32
Dropout	0.1	0.0	0.0	0.0	0.0	0.0
Batch Size	256	256	512	128	512	256
Iterations	varies*	15K	95K	80K	40K	20K
Learning Rate	5e-5					

Table 5. **Hyperparameters for voxel latent diffusion models.** *We train our model with 25K iterations for ShapeNet Airplane, 45K iterations for ShapeNet Car, and 35K iterations for ShapeNet Chair. For the Karton City dataset, we used the same hyperparameters as the Waymo dataset and trained the models to converge.