POINTSTAR: CONTINUOUS NEXT-SCALE AUTOREGRESSIVE GENERATION FOR 3D POINT CLOUD

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive point cloud generation is complicated by two properties of 3D geometry: point sets have no canonical ordering, and their coordinates are continuous. Existing methods address the first issue through manually ordered point sequences or coarse-to-fine next-scale prediction, but the latter commonly relies on vector-quantized tokens and therefore introduces a discrete bottleneck. We present PointSTAR, which formulates point cloud generation as continuous next-scale autoregressive prediction. PointSTAR first learns a compact hierarchy of progressively decodable continuous latents with a Nested-scale Variational AutoEncoder (NAVE). It then models generation at two complementary levels: inter-scale prediction captures coarse-to-fine structural evolution, while a multi-directional autoregressive flow captures dependencies among latent slots within each scale. This formulation preserves an explicit generation hierarchy without imposing a codebook on continuous geometry. Experiments on ShapeNet report competitive generation quality together with runtime comparisons, while reconstruction, interpolation, and component analyses examine the learned continuous hierarchy. PointSTAR also adapts to point cloud completion and upsampling, suggesting that the hierarchy is useful beyond unconditional generation.

1 INTRODUCTION

Point clouds represent 3D surfaces as unordered samples in continuous Euclidean space and are widely used in shape generation, reconstruction, computer-aided design, and robotic perception (Guo et al., 2021; Yang et al., 2019). Their set structure avoids committing to a regular grid, but it also removes the canonical order that sequence models rely on. At the same time, small coordinate changes can encode meaningful geometric detail. A point cloud generator must therefore respect unorderedness without discarding the continuity of the underlying geometry (Qi et al., 2017a;b; Zaheer et al., 2017).

Diffusion models address this challenge by learning point cloud distributions directly in continuous spaces and have achieved strong generation quality (Luo & Hu, 2021; Zhou et al., 2021; Zeng et al., 2022; Ren et al., 2024). Their generation process, however, is organized primarily as iterative denoising rather than as an explicit progression from global structure to local geometry. Autoregressive models offer a complementary perspective: they expose conditional dependencies between already generated structure and subsequent predictions. Applying this principle to point clouds is nontrivial because a point-wise autoregressive factorization requires an artificial serialization of an unordered set (Sun et al., 2020; Cheng et al., 2022; Chen et al., 2023).

Next-scale prediction provides a more natural autoregressive unit. Instead of generating individual points in a prescribed order, PointNSP predicts a hierarchy of increasingly detailed shape scales (Meng et al., 2026). This coarse-to-fine factorization makes structural evolution explicit and avoids treating raw points as a canonical sequence. Nevertheless, the hierarchy is represented with vector-quantized tokens: continuous geometric features must first be mapped to entries in a finite codebook. The resulting quantization bottleneck can discard geometric variation before the autoregressive model sees it. Recent visual generative models show that autoregressive prediction need not be restricted to discrete tokens and can instead model continuous representations directly (Li et al., 2024; Zheng et al., 2025; Han et al., 2026).

We therefore ask: *Can next-scale point cloud generation remain fully continuous while retaining an explicit coarse-to-fine autoregressive structure?* Answering this question requires solving two coupled problems. First, intermediate encoder features are not automatically suitable generation scales: each latent scale should be compact, and every successive prefix should decode into a complete shape at a distinct level of fidelity. Second, replacing codebook classification with continuous prediction requires modeling a high-dimensional conditional density, including dependencies both across scales and among latent slots within the same scale.

To address the representation problem, we introduce a Nested-scale Variational AutoEncoder (NAVE). NAVE compresses a point cloud into a hierarchy of continuous latent slots and applies progressive generative supervision so that every latent prefix reconstructs a complete shape, from coarse structure to fine geometry. Unlike generic hierarchical encoder features, these latents are explicitly trained to be progressively decodable and thus provide well-defined prediction targets for next-scale generation. To address the distribution-modeling problem, we introduce PointSTAR, a continuous autoregressive flow operating on the NAVE hierarchy. Across scales, PointSTAR predicts each finer representation conditioned on the generated coarser representations. Within a scale, it factorizes the joint density of latent slots and varies their causal order across flow blocks, exposing each slot to complementary contexts rather than a single fixed ordering. Together, these two levels of factorization turn unordered point cloud generation into structured autoregression without discretizing its geometry.

Our experiments evaluate the complete argument in stages. Reconstruction and interpolation analyses examine whether NAVE learns compact and continuous scale representations. Generation comparisons evaluate sample quality and efficiency. Component ablations isolate progressive representation learning, cross-scale conditioning, and multi-directional intra-scale modeling. Finally, completion and upsampling experiments test whether the learned hierarchy can be adapted beyond unconditional generation.

Our main contributions are: (i) We formulate point cloud generation as *continuous next-scale autoregressive prediction*, preserving an explicit coarse-to-fine generation process without point serialization or vector quantization. (ii) We introduce NAVE, a progressively supervised variational autoencoder that learns compact continuous latents from which complete shapes can be decoded at increasing geometric fidelity. (iii) We develop PointSTAR, which combines inter-scale conditioning with multi-directional intra-scale autoregressive flows to model the joint distribution of the continuous latent hierarchy. (iv) We thoughly evaluate our proposed representation and modeling choices, and adapt the framework to point cloud completion and upsampling.

2 Related Work

Point Cloud Generative Modeling Early methods learn point cloud distributions through latent-space models, adversarial training, or hierarchical set representations (Achlioptas et al., 2018; Shu et al., 2019; Kim et al., 2021). Flow-based methods, including PointFlow, DPF-Net, and SoftFlow, model continuous shape and point distributions (Yang et al., 2019; Klokov et al., 2020; Kim et al., 2020), while ShapeGF learns gradient fields for shape generation (Cai et al., 2020). Diffusion models operate on coordinates, point–voxel features, or latent representations (Luo & Hu, 2021; Zhou et al., 2021; Zeng et al., 2022), with subsequent work incorporating time-varying, frequency-aware, and structural mechanisms (Ren et al., 2024; Zhou et al., 2024; 2026). Straight-flow and transport-based methods further explore efficient continuous generation (Wu et al., 2023; Hui et al., 2025). These approaches motivate continuous geometric modeling. PointSTAR organizes the latent distribution into an explicit coarse-to-fine autoregressive hierarchy.

Autoregressive and Next-Scale Generation Autoregressive point cloud models require an ordering of their prediction units. PointGrow, CanonicalVAE, ShapeFormer, and PointGPT use coordinate, canonical, sparse, or patch-based organizations (Sun et al., 2020; Cheng et al., 2022; Yan et al., 2022; Chen et al., 2023). Visual Autoregressive Modeling (VAR) shifts the prediction unit from individual tokens to resolution scales (Tian et al., 2024). PointNSP applies next-scale prediction to point clouds using vector-quantized level-of-detail tokens (Meng et al., 2026). PointSTAR retains this scale-wise factorization but predicts continuous latent scales, avoiding the finite-codebook bottleneck. Each latent prefix is trained to decode a complete shape, so successive scales provide explicit stages of geometric refinement.

Continuous Representations and Autoregressive Flows Vector quantization enables token classification but introduces a trade-off between compression and quantization error (van den Oord et al., 2017; Esser et al., 2021). MAR and NOVA demonstrate autoregressive generation with continuous visual latents (Li et al., 2024; Deng et al., 2025), while DisCon, NextStep, and TokenBridge further investigate continuous prediction and its relationship to discrete tokens (Zheng et al., 2025; Han et al., 2026; Wang et al., 2025). For point clouds, hierarchical encoders such as PointNet++ and Point-M2AE learn features at multiple resolutions (Qi et al., 2017b; Zhang et al., 2022), but do not explicitly train each latent prefix to reconstruct a complete shape. LoST emphasizes increasing semantic detail in 3D token hierarchies (Dutt et al., 2026). NAVE provides progressively decodable continuous scales through reconstruction supervision at every stage. To model their distribution, PointSTAR builds on TARFlow, which combines causal Transformers with normalizing flows and alternates forward and reverse orders (Zhai et al., 2025). PointSTAR uses multiple complementary causal orders within each latent scale and conditions finer scales on coarser representations, combining continuous density modeling with explicit next-scale prediction.

3 METHOD

Our proposed method of continuous next-scale 3D point cloud generation can be defined into two coupled stages: learning a hierarchy of valid continuous prediction targets and modeling the joint distribution of that hierarchy. Figure 1 gives an overview. We first formalize continuous next-scale prediction in Sec. 3.1. Section 3.2 introduces NAVE, which learns compact latent scales that are progressively decodable into complete point clouds. Section 3.3 then presents PointSTAR, which combines inter-scale conditioning with multi-directional autoregressive flows within each scale (see Appendix B for details).

3.1 CONTINUOUS NEXT-SCALE FORMULATION

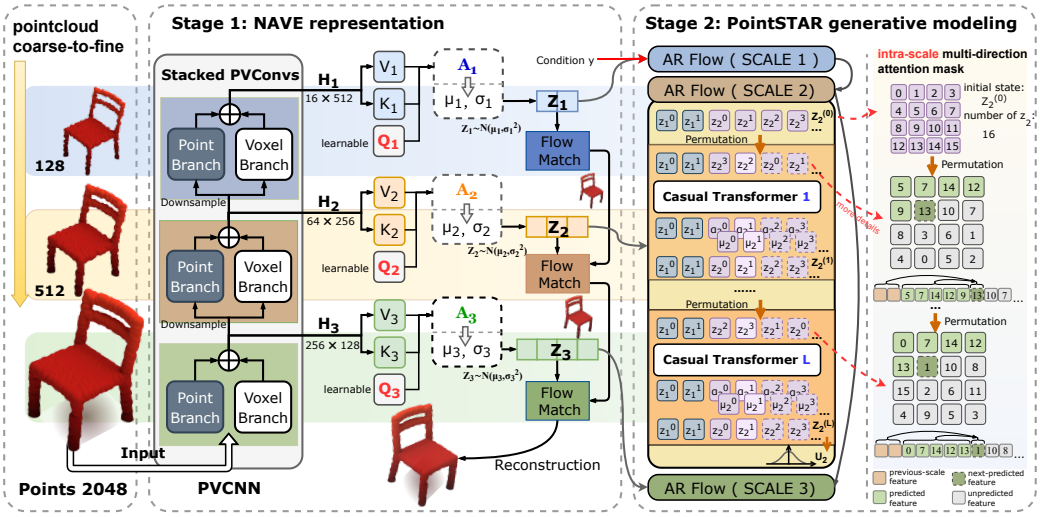

Figure 1: Overview of PointSTAR. NAVE learns progressively decodable continuous latent scales, and PointSTAR generates these scales from coarse structure to fine geometry while autoregressively modeling the latent slots within each scale.

Let $X_0 \in \mathbb{R}^{N\times 3}$ denote an input point cloud. We seek a latent hierarchy $\mathcal{Z} = \{Z_1, \ldots, Z_K\}$ ordered from coarse to fine, where $Z_k = \{z_{k,1}, \ldots, z_{k,M_k}\}$ contains M_k continuous latent slots and $z_{k,i} \in \mathbb{R}^{D_k}$. Every prefix $Z_{1:k}$ is trained to decode into a complete point cloud X_k at a corresponding resolution. The hierarchy therefore defines explicit generation stages rather than merely intermediate encoder activations. Formally, the latent distribution along the scale hierarchy is factorized as:

$$p_\theta(\mathcal{Z}) = \prod_{k=1}^{K} p_\theta(Z_k \mid Z_{1:k-1}), \tag{1}$$

where $Z_{1:0} = \varnothing$ for the first scale. This factorization replaces point-wise serialization with a coarse-to-fine generation order. In contrast to vector-quantized next-scale methods (Meng et al., 2026), the variables in Equation 1 remain continuous and introduce no finite codebook. Inter-scale conditioning alone does not capture dependencies among the slots of Z_k. Given a causal ordering π of the M_k slots, we further factorize the conditional distribution within each scale:

$$p_\theta(Z_k \mid Z_{1:k-1}; \pi) = \prod_{i=1}^{M_k} p_\theta\big(z_{k,\pi(i)} \mid Z_{1:k-1}, z_{k,\pi(1:i-1)}\big)\,. \tag{2}$$

Equation 1 and 2 expose the two dependencies modeled by PointSTAR, i.e., preceding scales provide global structural context, while preceding slots under π provide context within the current scale.

3.2 Progressively Decodable Representation with NAVE

A continuous next-scale model requires more than a generic multi-resolution feature pyramid. Ieadly, its latent variables must be compact enough for density modeling, yet each successive latent prefix must retain sufficient information to recover a complete shape. We construct these prediction targets with a Nested-scale Variational AutoEncoder (NAVE), consisting of a hierarchical encoder, a cascaded latent prior, and a progressive flow-matching decoder.

3.2.1 Hierarchical Continuous Encoder

Our NAVE encoder extracts a hierarchy of progressively coarser latent scales from the input point cloud. Each scale is represented by a fixed number of continuous latent slots. In our case, the widely-used PVCNN (Liu et al., 2019; Zhou et al., 2021; Zeng et al., 2022), which combines point-wise features with voxel convolutions, is employed as the backbone to extract point-voxel features at progressively coarser spatial resolutions:

$$\{H_1, \ldots, H_K\} = E_{\mathrm{PVCNN}}(X_0). \tag{3}$$

For scale k, M_k learnable queries Q_k aggregate the corresponding feature map via cross-attention,

$$A_k = \mathrm{CrossAttn}(Q_k, H_k, H_k), \qquad A_k \in \mathbb{R}^{M_k \times D_k}. \tag{4}$$

The number and width of the queries explicitly control latent capacity, which balances reconstruction fidelity against the difficulty of subsequent continuous density modeling (Yao et al., 2025). Then, two prediction heads parameterize a diagonal Gaussian posterior from A_k:

$$q_\phi(Z_k \mid X_0) = \mathcal{N}\big(\mu_k, \mathrm{diag}(\sigma_k^2)\big)\,, \qquad Z_k = \mu_k + \sigma_k \odot \epsilon_k \in \mathbb{R}^{M_k \times D_k}, \quad \epsilon_k \sim \mathcal{N}(0, I). \tag{5}$$

This yields the continuous hierarchy $\mathcal{Z} = E_\phi(X_0) = \{Z_1, \ldots, Z_K\}$. We use a standard Gaussian prior for Z_1 and a cascaded conditional prior for finer scales:

$$p_\omega(\mathcal{Z}) = p(Z_1) \prod_{k=2}^{K} p_\omega(Z_k \mid Z_{k-1}). \tag{6}$$

The prior regularizes adjacent scales without forcing the latents into a discrete codebook.

3.2.2 Progressive Flow-Matching Decoder

Hierarchical encoding alone does not make intermediate latents valid generation scales. NAVE therefore decodes every prefix $Z_{1:k}$ into a complete point cloud X_k, with point counts increasing from coarse to fine. At scale k, the decoder uses the newly introduced Z_k as its semantic condition and, for $k > 1$, the preceding reconstruction $\hat{X}_{k-1}$ as its geometric condition. Thus a finer stage refines an already complete global structure instead of predicting an isolated local residual. Specifically, we implement each stage as a conditional flow-matching decoder. Given target X_k, Gaussian noise $\xi_k \sim \mathcal{N}(0, I)$, and $t \sim \mathcal{U}(0, 1)$, the linear probability path and target velocity are

$$X_{k,t} = (1 - t)\xi_k + tX_k, \qquad v_k^* = X_k - \xi_k. \tag{7}$$

The scale-wise objective is then defined as the expected squared error between the predicted and target velocities:

$$\mathcal{L}_{\mathrm{FM}}^{(k)} = \mathbb{E}_{t,\xi_k}\left[\left\| v_{\psi_k}(X_{k,t}, t \mid Z_k, \hat{X}_{k-1}) - v_k^* \right\|_2^2\right], \tag{8}$$

where the geometric condition is omitted for $k = 1$. Integrating the learned velocity fields produces a sequence of complete reconstructions $\hat{\mathcal{X}} = \{\hat{X}_1, \ldots, \hat{X}_K\}$. Finally, NAVE jointly optimizes progressive reconstruction and variational regularization:

$$\mathcal{L}_{\text{NAVE}} = \sum_{k=1}^{K} \beta_k \mathcal{L}_{\text{FM}}^{(k)} + \sum_{k=1}^{K} \lambda_k D_{\text{KL}}[q_\phi(Z_k \mid X_0) \,\|\, p_\omega(Z_k \mid Z_{k-1})]\,. \tag{9}$$

Progressive supervision gives each latent scale an explicit geometric meaning. Once trained, its encoder and decoder are frozen while PointSTAR learns the distribution of the resulting hierarchy.

3.3 Continuous Next-Scale Autoregressive Flow

Given the NAVE hierarchy of continuous latents, PointSTAR thus models the conditional distributions in Equation 1 with Transformer-based autoregressive normalizing flows. For scale k, an invertible transformation maps the current latent to a base variable and supports generation through its inverse:

$$U_k = F_{\theta,k}(Z_k \mid Z_{1:k-1}, t), \qquad Z_k = F_{\theta,k}^{-1}(U_k \mid Z_{1:k-1}, t). \tag{10}$$

For $k > 1$, the preceding scales parameterize a conditional Gaussian base $p_0(U_k \mid Z_{1:k-1})$ and are also provided as a structural prefix to every autoregressive block. For $k = 1$, the base distribution is learned without cross-scale context. Then the conditional log likelihood is defined as:

$$\log p_\theta(Z_k \mid Z_{1:k-1}, t) = \log p_0(U_k \mid Z_{1:k-1}) + \log \left| \det \frac{\partial F_{\theta,k}}{\partial Z_k} \right|. \tag{11}$$

Multi-directional intra-scale modeling. Each flow block l applies a causal ordering π_l to the current-scale slots. A causal Transformer predicts the affine parameters for a slot from the preceding scales and the slots that appear earlier under that ordering:

$$(\mu_{k,l}^i, \alpha_{k,l}^i) = h_{\theta,l}\big(Z_{1:k-1}, z_{k,\pi_l(1:i-1)}, t\big)\,. \tag{12}$$

Class-conditional generation can be supported by additionally conditioning $h_{\theta,l}$ on a learnable category embedding e_y associated with the class label y. $(\mu_{k,l}^i, \alpha_{k,l}^i)$ can be used to produce $z_{k,l}^i$, which can construct Z_k then decode as a whole point cloud. The corresponding autoregressive affine transformation is invertible and its Jacobian is triangular, enabling exact likelihood evaluation. A fixed ordering would repeatedly assign every slot the same conditional role. Inspired by the alternating directions in TARFlow (Zhai et al., 2025), we instead distribute complementary orderings across flow blocks. Consequently, slots occupy different causal positions across depth and are modeled under more diverse intra-scale contexts.

Multi-noise training and iterative refinement. To further improve robustness in the continuous latent space, we follow the multi-noise training strategy of iTARFlow (Chen et al., 2026). At scale k, we perturb the encoded latent with a continuously sampled noise level:

$$\widetilde{Z}_{k,t} = Z_k + t\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \quad t \sim \mathcal{U}(t_{\min}, t_{\max}^{(k)}). \tag{13}$$

Then the time-conditioned flow minimizes a weighted negative log likelihood,

$$\mathcal{L}_{\text{STAR}} = -\sum_{k=1}^{K} \mathbb{E}\Big[\gamma_k(t) \log p_\theta(\widetilde{Z}_{k,t} \mid Z_{1:k-1}, t)\Big]\,, \tag{14}$$

where $\gamma_k(t)$ controls the contribution of each noise level. During inference, PointSTAR first samples a latent at $t_{\max}^{(k)}$ through the inverse flow, then refines it toward lower noise levels using the score obtained from the explicit likelihood:

$$\mathrm{d}Z_{k,t} = -t\, \nabla_{Z_{k,t}} \log p_\theta(Z_{k,t} \mid Z_{1:k-1}, t)\, \mathrm{d}t. \tag{15}$$

Generation proceeds recursively from $\hat{Z}_1$ to $\hat{Z}_K$. Each sampled scale is appended to the context for subsequent scales, producing $\hat{\mathcal{Z}} = \{\hat{Z}_1, \ldots, \hat{Z}_K\}$. The frozen NAVE decoder then maps this hierarchy to progressively refined point clouds:

$$\hat{X}_k = D_{\psi_k}(\hat{Z}_k, \hat{X}_{k-1}), \qquad k = 1, \ldots, K, \tag{16}$$

with the preceding geometry omitted at the first scale. The final output is the finest generation $\hat{X}_K$.

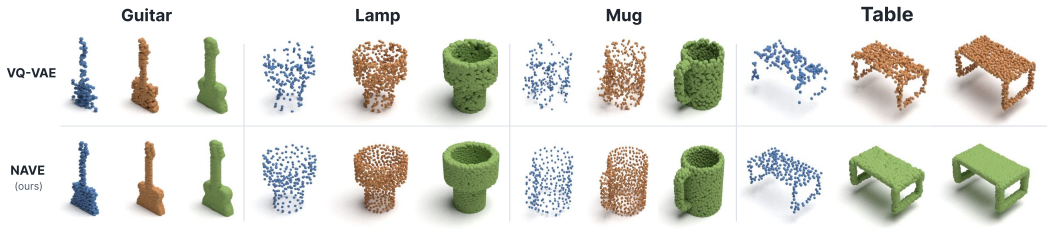

Figure 2: Progressive reconstruction by PointNSP's VQ-VAE (top) and NAVE (bottom).

4 Experiments

In this section, we evaluate NAVE's reconstruction and progressive decoding, PointSTAR's generation quality and cost, and adaptation to completion and upsampling.

4.1 Experimental Setup

Datasets. Following PointFlow (Yang et al., 2019), we use ShapeNetV2 with 2048 points per shape and global dataset-level normalization. Airplane, Chair, and Car have training/evaluation splits of $2832/405$, $4612/662$, and $2458/352$ shapes, respectively. We also evaluate a model trained across 55 categories, similar to PointNSP Meng et al. (2026).

Evaluation metrics. Following PVD (Zhou et al., 2021) and LION (Zeng et al., 2022), we evaluate generation using Coverage (COV), Minimum Matching Distance (MMD), and 1-nearest-neighbor accuracy (1-NNA) under Chamfer Distance (CD) and Earth Mover's Distance (EMD). Higher COV, lower MMD, and 1-NNA closer to 50% are better. We report CD/EMD for reconstruction, completion, and upsampling, and GPU hours and sampling time for efficiency.

Implementation details. NAVE uses $1/4/16$ latent slots of dimensions $512/256/128$ and decodes $128/512/2048$ points across three scales. We train it with AdamW at 1×10^{-4} for 200k iterations with batch size 32. PointSTAR uses $8/16/32$ flow blocks, noise intervals $[0.01, 0.2]$, $[0.01, 0.3]$, and $[0.01, 0.5]$, and ten refinement steps per scale. It is trained with AdamW at 2×10^{-4} for 50k iterations with batch size 32. All training uses one NVIDIA RTX 3090 with 24 GB memory. Appendix A collects the reported settings, evaluation scope, and runtime and downstream-task details.

4.2 NAVE Reconstruction and Progressive Decoding

We compare NAVE with PointFlow (Yang et al., 2019), SoftFlow (Kim et al., 2020), ShapeGF (Cai et al., 2020), Canonical VAE (Cheng et al., 2022), IP-Encoder (Röell & Rieck, 2025), and PointNSP's VQ-VAE (Meng et al., 2026). Table 1 reports 2048-point reconstruction, with full means and standard deviations in Appendix Table 8. NAVE achieves the lowest reported CD on all three categories and the lowest EMD on Chair. It outperforms PointNSP's VQ-VAE in both metrics, while IP-Encoder has lower EMD on Airplane and Car.

Table 1: Reconstruction (CD / EMD (↓)) at 2048 pts.

Method	Airplane	Chair	Car
PointFlow	1.30 / 5.36	10.43 / 17.54	6.94 / 12.93
SoftFlow	1.19 / 4.28	11.05 / 17.68	6.82 / 11.44
ShapeGF	1.05 / 4.42	5.96 / 12.23	5.68 / 9.26
Canonical VAE	0.98 / 3.19	6.56 / 8.60	5.44 / 6.13
IP-Encoder	1.05 / **1.57**	9.24 / 6.19	5.82 / **3.18**
PointNSP (VQ-VAE)	1.02 / 2.71	5.24 / 6.35	5.31 / 4.92
NAVE (Ours)	**0.96** / 2.42	**4.97** / **5.93**	**5.12** / 4.67

Figure 2 compares progressive decoding on Guitar, Lamp, Mug, and Table. Each NAVE prefix $Z_{1:k}$ reconstructs a complete shape at 128, 512, or 2048 points, with more coherent intermediate geometry than PointNSP's VQ-VAE in these examples. Successive scales refine global structure with finer detail, making each prefix a meaningful generation target for Sec. 3.1.

4.3 NAVE Ablation Studies

Encoder ablation. Figure 3(a) compares PVCNN, DGCNN, PointNet, and Point Transformer. We can see that PVCNN offers the best balance of reconstruction and downstream generation and is used throughout the remaining evaluations.

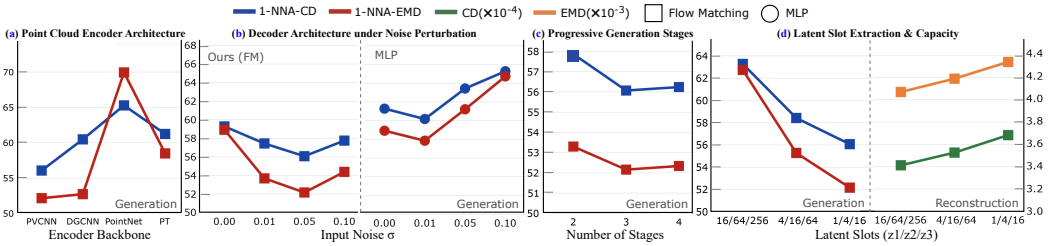

Figure 3: NAVE ablations on (a) encoder backbone, (b) decoder robustness, (c) progressive stage count, and (d) latent slot allocation.



Figure 4: Latent interpolation decoded with the frozen NAVE decoder.

Decoder robustness. Generated latents may differ from encoded ones. We adopt a flow-matching decoder to accommodate the additional uncertainty in continuous latent modeling. Figure 3(b) compares the flow-matching decoder with an MLP under increasing latent perturbations. The flow-matching decoder better preserves reconstruction quality and yields better downstream generation, supporting its robustness to generation errors.

Latent scales. Figure 3(c) compares two, three, and four progressive stages. Three improve reconstruction and generation over two, while a fourth brings little reconstruction gain and worsens generation. To isolate hierarchy from slot count, Table 2(A) fixes 21 slots and the intra-scale ordering. Increasing from one to three scales improves CD/EMD 1-NNA from 60.43/58.27 to 56.07/52.16, supporting the three-scale hierarchy.

Table 2: Latent hierarchy and representation ablations. (A) varies the allocation of 21 latent slots. (B) compares continuous and quantized latents at fixed latent unit count N or codebook size K.

A: Latent hierarchy

Scales	Slots	1-NNA CD	1-NNA EMD
1	21	60.43	58.27
2	$5+16$	57.61	54.38
3	$\mathbf{1+4+16}$	**56.07**	**52.16**

All variants in (A) use multi-directional intra-scale ordering.

K: codebook size; N: latent units. Q. err.: quantization error. 1-NNA is better closer to 50%.

B: Continuous and quantized latents

Variant	Representation K	N	Usage (%)	Q. err. (↓)	Recon. (↓) CD	Recon. (↓) EMD	1-NNA (→ 50%) CD	1-NNA (→ 50%) EMD
Continuous	–	**21**	–	–	**3.68**	**4.34**	**56.07**	**52.16**
Fixed latent units $N = 21$								
Quantized	256	21	17.63	0.0451	6.47	10.28	64.38	60.27
Quantized	**512**	**21**	10.84	0.0386	**5.22**	**8.94**	**60.62**	**57.49**
Quantized	1024	21	6.37	0.0354	5.18	9.89	61.31	58.16
Quantized	2048	21	3.72	**0.0332**	6.05	9.97	62.57	60.48
Fixed codebook size $K = 512$								
Quantized	512	84	16.57	0.0264	3.76	5.91	63.18	59.07
Quantized	512	336	**23.44**	**0.0179**	**3.38**	**4.13**	65.91	62.83

Latent capacity. Figure 3(d) varies slot counts. Larger capacity improves reconstruction but makes generation harder. The selected $1/4/16$ slots with widths $512/256/128$ provide the best reported balance. Capacity depends on both slot count and width (see Appendix C.2).

Continuous vs. quantized latents. Table 2(B) compares representations at fixed latent unit count $N = 21$, then increases quantized capacity at codebook size $K = 512$. At fixed N, continuous latents achieve lower reconstruction errors and 1-NNA closer to 50%. Increasing K reduces quantization error but lowers codebook usage, without consistent generation gains. At $N = 336$, quantized latents improve reconstruction CD/EMD to 3.38/4.13 versus 3.68/4.34 for the continuous model, but worsen generation 1-NNA to 65.91/62.83 versus 56.07/52.16. Greater quantized capacity thus improves reconstruction without making generation easier. The fixed-N comparison controls unit count, while larger-N variants trade additional capacity for reconstruction.

Continuity. We interpolate between encoded shapes and decode intermediate latents with the frozen NAVE decoder. Figure 4 shows recognizable within-category and cross-category transitions, providing a qualitative check of latent continuity. Additional within-category and cross-category examples appear in Appendix D.5, including Figure 19.

Table 3: Single-category unconditional generation results. PointSTAR's training cost includes NAVE and latent-generator training, inference uses ten refinement steps per scale.

		Airplane						Chair						Car						Efficiency	
		MMD (↓)		COV (↑)		1-NNA (→ 50)		MMD (↓)		COV (↑)		1-NNA (→ 50)		MMD (↓)		COV (↑)		1-NNA (→ 50)		Training (↓)	Inference (↓)
Class	**Model**	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD	GPU h	s
Normalizing Flow	PointFlow	0.224	0.390	47.90	46.41	75.68	70.74	2.409	1.595	42.90	50.00	62.84	60.57	0.901	0.807	46.88	50.00	58.10	56.25	360	1.73
	DPF-Net	0.264	0.409	46.17	48.89	75.18	65.55	2.536	1.632	44.71	48.79	62.00	58.53	1.129	0.853	45.74	49.43	62.35	54.48	—	—
	SoftFlow	0.231	0.375	46.91	47.90	76.05	65.80	2.528	1.682	41.39	47.43	59.21	60.05	1.187	0.859	42.90	44.60	64.77	60.09	—	—
Diffusion	ShapeGF	0.270	0.660	40.74	40.49	80.00	76.17	2.89	1.700	46.47	48.03	68.96	65.48	0.923	0.760	49.43	50.28	63.20	56.53	**28**	20.60
	DPM	0.213	0.572	48.64	33.83	76.42	86.91	2.399	2.066	44.86	35.50	60.05	74.77	0.902	1.140	44.03	34.94	68.89	79.97	—	—
	PVD	0.224	0.370	48.88	52.09	73.82	64.81	2.622	1.556	49.84	50.60	56.26	53.32	1.077	0.794	41.19	50.56	54.55	53.83	142	8.46
	LION	0.219	0.372	47.16	49.63	67.41	61.23	2.640	1.550	48.94	52.11	53.70	52.34	0.913	0.752	50.00	**56.53**	53.41	51.14	550	27.12
	TIGER		——			67.21	56.26		——			54.32	51.71		——			54.12	50.24	164	9.73
	FrePoLad		——			65.25	62.10		——			52.35	53.23		——			51.89	50.26	—	—
	Quartet		——			**63.3**	59.70		——			**50.10**	51.8		——			51.6	53.7	50	5.45
Flow Matching	PSF		——			71.11	61.09		——			58.92	54.45		——			57.19	56.07	—	—
	not-so-ot-flow		——			68.84	61.85		——			55.51	57.63		——			59.66	53.35	—	—
Auto-regressive	PointGrow	3.07	11.64	10.62	10.62	99.38	99.38	16.23	18.83	12.08	13.75	98.05	99.10	14.12	18.33	6.82	11.65	99.86	98.01	—	—
	CanonicalVAE	0.83	0.550	45.67	44.19	63.45	71.60	7.37	1.15	45.77	46.07	60.12	61.93	3.31	0.889	41.76	47.72	55.68	57.81	142	5.45
	MAR	0.219	0.368	47.90	49.14	75.06	65.80	2.579	1.556	41.09	46.98	63.82	60.80	1.444	0.900	37.50	42.90	65.77	61.36	—	—
	PointNSP		——			66.98	56.05		——			54.01	53.76		——			53.12	**50.09**	178	3.59
	PointSTAR	0.209	**0.338**	**50.86**	**53.57**	65.19	**54.44**	**2.329**	**1.478**	**50.45**	**53.17**	52.45	**51.06**	0.873	0.7441	**51.14**	53.13	**50.57**	50.99	46	**1.37**

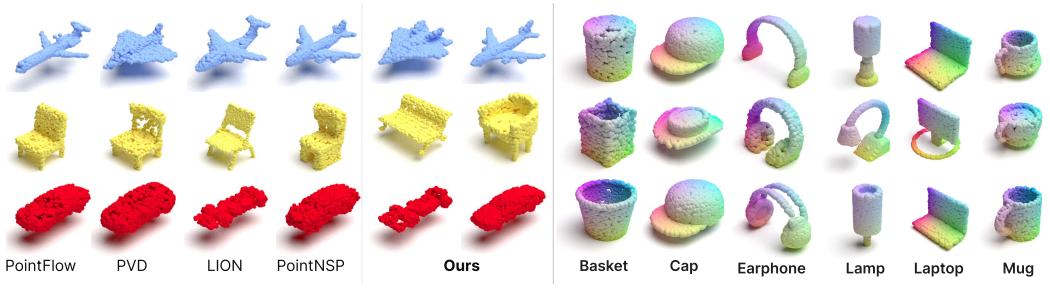

Figure 5: **Left:** Single-class generation comparisons. **Right:** Our generated multi-class results.

4.4 Point Cloud Generation

Baselines. We compare with PointFlow (Yang et al., 2019), SoftFlow (Kim et al., 2020), DPM (Luo & Hu, 2021), PVD (Zhou et al., 2021), LION (Zeng et al., 2022), TIGER (Ren et al., 2024), FrePoLad (Zhou et al., 2024), Quartet (Zhou et al., 2026), PointGrow (Sun et al., 2020), CanonicalVAE (Cheng et al., 2022), and PointNSP (Meng et al., 2026).

Single-category results. Table 3 evaluates one model per category. PointSTAR achieves the highest reported CD/EMD COV on Airplane and Chair and the lowest CD/EMD MMD on Chair. Its Chair and Car 1-NNA values approach 50%, while Airplane 1-NNA-CD (65.19) trails several baselines despite high coverage. The two-stage pipeline takes 46 GPU hours, including NAVE and generator training, and samples in 1.37 seconds with ten refinement steps per scale. Among methods with reported costs, it has the lowest inference time and second-lowest training time. Unspecified timing boundaries, including NAVE decoding, and batch settings limit strict cross-method comparisons. Figure 5 provides qualitative comparisons. Appendix D.1 visualizes progressive generation, and Appendix D.2 provides additional category-specific examples.

Multi-category results. Table 4 evaluates a model trained across 55 categories. PointSTAR achieves the lowest listed CD/EMD on Airplane and Chair and the lowest CD on Car. Its Car EMD (53.53) is slightly higher than PointNSP's (52.71). These results suggest that the hierarchy remains useful across diverse categories. Additional outputs appear in Appendix D.3.

Table 4: Multi-class generation results.

Model	**Airplane**		**Chair**		**Car**	
	CD ↓	EMD ↓	CD ↓	EMD ↓	CD ↓	EMD ↓
PVD	97.53	99.88	88.37	96.37	89.77	94.89
TIGER	83.54	81.55	57.34	61.45	65.79	57.24
PointGPT	94.94	91.73	71.83	79.00	89.35	87.22
LION	86.30	77.04	66.50	63.85	64.52	54.21
PointNSP	75.42	66.54	56.03	52.22	58.76	**52.71**
Ours	**74.81**	**64.98**	**54.42**	**51.80**	**56.79**	53.53

4.5 PointSTAR Ablation Studies

Inter-scale conditioning. Table 5(A) varies the context for generating Z_3 with multi-directional intra-scale modeling fixed. CD/EMD 1-NNA improves from 63.18/62.44 without guidance to 59.43/56.92 with Z_1 and 56.07/52.16 with $Z_{1:2}$. Both preceding scales provide the most effective context for finest-scale generation.

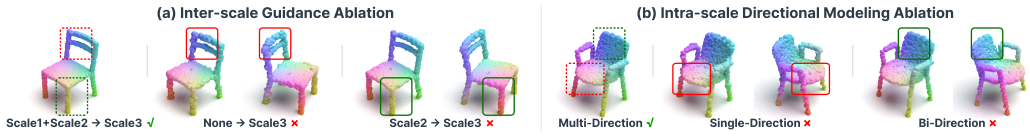

Figure 6: Ablations of cross-scale guidance and intra-scale ordering.

Autoregressive ordering. With $Z_{1:2}$ fixed, Table 5(B) compares causal orderings across flow blocks. Multi-directional ordering achieves 56.07/52.16, outperforming parallel, single-direction, bi-directional, and random multi-direction ordering. Its advantage over the random schedule (57.36/52.82) supports the proposed causal contexts in Eq. 2. Figures 6 and 8 (Appendix C.3) illustrate guidance and ordering.

Table 5: PointSTAR ablations on cross-scale guidance, intra-scale ordering, and noise refinement.

Setting	CD 1-NNA	EMD 1-NNA
A: Cross-scale guidance		
None	63.18	62.44
Z_1	59.43	56.92
$Z_{1:2}$ (Ours)	**56.07**	**52.16**
B: Intra-scale ordering		
Parallel	64.24	62.17
Single-direction	61.28	57.84
Bi-direction	58.29	54.68
Random multi-direction	57.36	52.82
Multi-direction (Ours)	**56.07**	**52.16**

C: Noise training and refinement

Noise	Steps	1-NNA CD	1-NNA EMD	Time (s)
Small-scale	0	60.71	56.28	0.79
	1	59.13	53.22	1.02
Multi-scale	0	66.40	64.88	0.79
	1	61.33	59.79	0.83
	3	58.94	53.75	0.99
	5	57.56	52.87	1.09
	10 (Ours)	**56.07**	**52.16**	1.37

Multi-noise training and iterative refinement. Table 5(C) jointly ablates noise training and refinement. Without refinement, multi-scale noise yields worse CD/EMD 1-NNA than small-scale noise (66.40/64.88 vs. 60.71/56.28), and remains worse with one step. Three and ten multi-scale refinement steps improve 1-NNA to 58.94/53.75 and 56.07/52.16. Sampling time increases from 0.79 seconds without refinement to 0.99 and 1.37 seconds, respectively. The results support combining multi-noise training with sufficient refinement and show the quality–time trade-off.

4.6 Adaptation to Completion and Upsampling

PointSTAR follows PVD for completion and PointNSP for 2× upsampling (1024 to 2048 points). It achieves the lowest CD in all six category–task comparisons and the lowest EMD in five (Table 6); PointNSP has lower Car completion EMD (20.68 vs. 20.94). Qualitative examples appear in Figure 7 and Appendix D.4 (Figure 16).

Table 6: Results on point-cloud completion (**Left**) and upsampling (**Right**).

	Airplane		Chair		Car	
Model	**CD ↓**	**EMD ↓**	**CD ↓**	**EMD ↓**	**CD ↓**	**EMD ↓**
PVD	0.4415	10.30	3.211	29.39	1.774	21.46
PointNSP	0.4012	10.08	2.702	28.78	1.384	**20.68**
PointSTAR	**0.3985**	**10.05**	**2.684**	**28.49**	**1.352**	20.94

	Airplane		Chair		Car	
Model	**CD ↓**	**EMD ↓**	**CD ↓**	**EMD ↓**	**CD ↓**	**EMD ↓**
LION	0.7041	59.65	5.398	54.33	5.714	47.56
PointNSP	0.6889	58.86	5.204	51.03	5.585	46.74
PointSTAR	**0.6523**	**56.32**	**5.032**	**50.28**	**5.418**	**46.05**



Figure 7: Completion and upsampling with PVD, PointNSP, and PointSTAR.

5 Conclusion

In this work, we presented PointSTAR, a continuous next-scale autoregressive framework for 3D point cloud generation that preserves an explicit coarse-to-fine generation order and continuous geometry. NAVE provides progressively decodable latent scales, while PointSTAR models their distribution through inter-scale conditioning and multi-directional autoregressive factorization within each scale. Representation analyses, generation comparisons, and component ablations examine this design. Completion and upsampling further demonstrate adaptation to conditional tasks. The current work focuses on fixed-resolution object-level point clouds. Limitations are discussed in Appendix E. Future work includes variable-size point sets, large-scale scenes, additional attributes such as normals or colors, and adaptive scale decompositions that allocate latent capacity according to shape complexity while reducing continuous density modeling costs.

AI USE STATEMENT

We used generative AI tools to polish the writing of this paper, including improving grammar, wording, and readability. The authors reviewed and edited all AI-assisted text to ensure that it accurately reflects our work. We take full responsibility for the final content of this paper, including its text, claims, and results.

REFERENCES

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning representations and generative models for 3D point clouds. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 40–49, 2018.

Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *European Conference on Computer Vision*, pp. 364–381, 2020.

Guangyan Chen, Meiling Wang, Yi Yang, Kai Yu, Li Yuan, and Yufeng Yue. PointGPT: Auto-regressively generative pre-training from point clouds. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Tianrong Chen, Jiatao Gu, David Berthelot, Joshua Susskind, and Shuangfei Zhai. Normalizing flows with iterative denoising. In *Proceedings of the International Conference on Machine Learning*, 2026.

An-Chieh Cheng, Xueting Li, Sifei Liu, Min Sun, and Ming-Hsuan Yang. Autoregressive 3D shape generation via canonical mapping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 89–104, 2022.

Haoge Deng, Ting Pan, Haiwen Diao, Zhengxiong Luo, Yufeng Cui, Huchuan Lu, Shiguang Shan, Yonggang Qi, and Xinlong Wang. Autoregressive video generation without vector quantization. In *International Conference on Learning Representations*, 2025.

Niladri Shekhar Dutt, Zifan Shi, Paul Guerrero, Chun-Hao Paul Huang, Duygu Ceylan, Niloy J. Mitra, and Xuelin Chen. LoST: Level of semantics tokenization for 3D shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19950–19959, 2026.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12873–12883, 2021.

Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3D point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4338–4364, 2021.

Chunrui Han, Guopeng Li, Jingwei Wu, Quan Sun, Yan Cai, Yuang Peng, Zheng Ge, Deyu Zhou, Haomiao Tang, Hongyu Zhou, Kenkun Liu, Shu-Tao Xia, Binxing Jiao, Daxin Jiang, Xiangyu Zhang, and Yibo Zhu. NextStep-1: Toward autoregressive image generation with continuous tokens at scale. In *International Conference on Learning Representations*, 2026.

Ka-Hei Hui, Chao Liu, Xiaohui Zeng, Chi-Wing Fu, and Arash Vahdat. Not-so-optimal transport flows for 3D point cloud generation. In *International Conference on Learning Representations*, 2025.

Hyeongju Kim, Hyeonseung Lee, Woo Hyun Kang, Joun Yeop Lee, and Nam Soo Kim. SoftFlow: Probabilistic framework for normalizing flow on manifolds. In *Advances in Neural Information Processing Systems*, pp. 16388–16397, 2020.

Jinwoo Kim, Jaehoon Yoo, Juho Lee, and Seunghoon Hong. SetVAE: Learning hierarchical composition for generative modeling of set-structured data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15059–15068, 2021.

Roman Klokov, Edmond Boyer, and Jakob Verbeek. Discrete point flow networks for efficient point cloud generation. In *European Conference on Computer Vision*, pp. 694–710, 2020.

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

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 (CVPR)*, pp. 2837–2845, 2021.

Ziqiao Meng, Qichao Wang, Zhiyang Dou, Zixing Song, Zhipeng Zhou, Irwin King, and Peilin Zhao. PointNSP: Autoregressive 3D point cloud generation with next-scale level-of-detail prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 31450–31461, 2026.

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 (CVPR)*, pp. 652–660, 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5099–5108, 2017b.

Zhiyuan Ren, Minchul Kim, Feng Liu, and Xiaoming Liu. TIGER: Time-varying denoising model for 3D point cloud generation with diffusion process. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9462–9471, 2024.

Ernst Röell and Bastian Rieck. Point cloud synthesis using inner product transforms. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3D point cloud generative adversarial network based on tree structured graph convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3859–3868, 2019.

Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. PointGrow: Autoregressively learned point cloud generation with self-attention. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 61–70, 2020.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, pp. 6306–6315, 2017.

Yuqing Wang, Zhijie Lin, Yao Teng, Yuanzhi Zhu, Shuhui Ren, Jiashi Feng, and Xihui Liu. Bridging continuous and discrete tokens for autoregressive visual generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18596–18605, 2025.

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

Xingguang Yan, Liqiang Lin, Niloy J. Mitra, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. ShapeFormer: Transformer-based shape completion via sparse representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6239–6249, 2022.

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 (ICCV)*, pp. 4541–4550, 2019.

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15703–15712, 2025.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, volume 30, pp. 3391–3401, 2017.

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*, volume 35, pp. 10021–10039, 2022.

Shuangfei Zhai, Ruixiang Zhang, Preetum Nakkiran, David Berthelot, Jiatao Gu, Huangjie Zheng, Tianrong Chen, Miguel Ángel Bautista, Navdeep Jaitly, and Joshua M. Susskind. Normalizing flows are capable generative models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 74348–74369, 2025.

Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-M2AE: Multi-scale masked autoencoders for hierarchical point cloud pretraining. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Peng Zheng, Junke Wang, Yi Chang, Yizhou Yu, Rui Ma, and Zuxuan Wu. Rethinking discrete tokens: Treating them as conditions for continuous autoregressive image synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17390–17400, 2025.

Chenliang Zhou, Fangcheng Zhong, Param Hanji, Zhilin Guo, Kyle Thomas Fogarty, Alejandro Sztrajman, Hongyun Gao, and A. Cengiz Oztireli. FrePolad: Frequency-rectified point latent diffusion for point cloud generation. In *European Conference on Computer Vision*, pp. 434–453, 2024.

Chenliang Zhou, Fangcheng Zhong, Weihao Xia, Albert Miao, Canberk Baykal, and Cengiz Oztireli. Quartet of diffusions: Structure-aware point cloud generation through part and symmetry guidance. In *International Conference on Learning Representations*, 2026.

Linqi Zhou, Yilun Du, and Jiajun Wu. 3D shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5826–5835, 2021.

A Implementation and Evaluation Details

This appendix supplements the method and experiments in Secs. 3 and 4.

A.1 Architecture and Two-Stage Training

Table 7 collects the reported settings. NAVE's $1/4/16$ latent slots have widths $512/256/128$ and flattened dimensions $512/1024/2048$. Slot counts, feature widths, and output resolutions describe different aspects of the model: the $1/4/16$ latent hierarchy is distinct from the $128/512/2048$ output point counts.

Table 7: Implementation details for NAVE and PointSTAR.

Component	Setting	Value
NAVE	Output resolutions	$128/512/2048$
NAVE	Latent slots per scale	$1/4/16$
NAVE	Latent dimensions	$512/256/128$
NAVE	Optimizer / learning rate	AdamW / 1×10^{-4}
NAVE	Batch size / iterations	32 / 200k
PointSTAR	Flow blocks per scale	8/16/32
PointSTAR	Noise ranges $[t_{\min}, t_{\max}^{(k)}]$	$[0.01, 0.2]/[0.01, 0.3]/[0.01, 0.5]$
PointSTAR	Refinement steps	$10/10/10$
PointSTAR	Optimizer / learning rate	AdamW / 2×10^{-4}
PointSTAR	Batch size / iterations	32 / 50k
Runtime	GPU model / number of GPUs	NVIDIA RTX 3090 (24 GB) / 1

NAVE jointly learns its encoder, cascaded prior, and progressive decoder using Eq. 9, then remains frozen while PointSTAR learns the encoded hierarchy distribution using Eq. 14. Generation uses PointSTAR followed by the frozen decoder; NAVE's regularization prior and PointSTAR's sampling distribution serve different roles.

A.2 Data and Evaluation Scope

Single-category experiments use globally normalized ShapeNetV2 point clouds with 2048 points. Training/evaluation splits are $2832/405$ for Airplane, $4612/662$ for Chair, and $2458/352$ for Car. The multi-category setting trains across 55 categories and is evaluated separately.

COV measures reference-set coverage, MMD measures distances to the nearest generated matches, and 1-NNA measures how readily generated and reference sets can be distinguished. Higher COV, lower MMD, and 1-NNA closer to 50% are better. Reconstruction and downstream tasks report CD and EMD, for which lower values are better. Qualitative grids illustrate individual outputs rather than estimating distributional coverage.

A.3 Runtime and Downstream Tasks

The reported 46 GPU hours cover NAVE and generator training. Sampling takes 1.37 seconds with ten latent refinement steps per scale. These steps update the sampled latents and are distinct from point-space decoder integration steps. Timing boundaries and batch settings remain unspecified in the current experimental record, limiting strict cross-method runtime comparisons.

Completion maps a partial input to a complete shape under the PVD task setting. Upsampling maps 1024 points to 2048 points ($r = 2$) under the PointNSP task setting. Results appear in Table 6 and Figure 7. Additional qualitative examples appear in Figure 16.

B Additional Method Details

B.1 Autoregressive Affine Blocks and Exact Likelihood

For a scale k, write $C_k = Z_{1:k-1}$ for the preceding-scale context, with $C_1 = \varnothing$. Let $Y^{(0)} = Z_k$ and let $Y^{(l)}$ denote the output of flow block l. Under the ordering π_l, the block predicts a shift $\mu^i_{k,l}$ and a log scale $\alpha^i_{k,l}$ from C_k, the noise level t, and the preceding input slots $Y^{(l-1)}_{\pi_l(1:i-1)}$. An affine block can be written as

$$Y^{(l)}_{\pi_l(i)} = \left(Y^{(l-1)}_{\pi_l(i)} - \mu^i_{k,l}\right) \odot \exp\left(-\alpha^i_{k,l}\right). \tag{17}$$

The affine parameters depend only on preceding inputs under π_l, so the Jacobian is triangular, with log absolute determinant

$$\log\left|\det J_{k,l}\right| = -\sum_{i=1}^{M_k}\sum_{d=1}^{D_k} \alpha^i_{k,l,d}. \tag{18}$$

Permutation matrices have absolute determinant one and add no log-determinant term. For L_k blocks, the conditional log likelihood is therefore

$$\log p_\theta(Z_k \mid C_k, t) = \log p_0(Y^{(L_k)} \mid C_k) + \sum_{l=1}^{L_k} \log\left|\det J_{k,l}\right|. \tag{19}$$

The inverse traverses blocks in reverse order and recovers positions sequentially, using

$$Y^{(l-1)}_{\pi_l(i)} = Y^{(l)}_{\pi_l(i)} \odot \exp\left(\alpha^i_{k,l}\right) + \mu^i_{k,l}. \tag{20}$$

Changing π_l changes latent causal roles while preserving invertibility. These orderings operate on latent units rather than input points.

B.2 Training and Sampling Procedures

The following procedures summarize Sec. 3.

Stage I: learn progressively decodable targets.

1. Encode a training point cloud X_0 into Gaussian posterior parameters for each scale and sample Z_k by reparameterization.
2. Construct targets X_k, sample times and point-space noise, and evaluate the velocity losses in Eq. 8.
3. Add the KL terms in Eq. 9 and update NAVE.

Stage II: learn the latent hierarchy distribution.

1. Freeze NAVE and obtain continuous latent targets for each training point cloud.
2. At every scale, sample t from its noise interval and construct $\widetilde{Z}_{k,t}$ using Eq. 13.
3. Condition on preceding encoded scales and update PointSTAR using Eq. 14.

Inference: generate scales from coarse to fine.

1. Initialize the preceding-scale context to the empty set.
2. Sample U_k conditioned on the context and apply the inverse flow at $t^{(k)}_{\max}$.
3. Refine toward $t_{\min}$ using Eq. 15. Append $\hat{Z}_k$ to the context and repeat for the next scale.
4. Decode the sampled hierarchy with the frozen NAVE decoder to obtain complete point clouds at 128, 512, and 2048 points.

Stage II uses encoded context; inference uses generated context. Refinement keeps the preceding scales fixed while updating the current latent.

C Full Quantitative Results and Ablation Scope

C.1 Full Reconstruction Results

Table 8 preserves the complete 2048-point reconstruction results, including the standard deviations omitted from the compact main-text Table 1. The PointNSP row evaluates its VQ-VAE representation. NAVE has the lowest listed CD in all three categories and the lowest listed EMD on Chair; IP-Encoder has lower EMD on Airplane and Car. The table therefore supports metric-specific comparisons rather than a uniform advantage under every distance.

Table 8: Full reconstruction results at 2048 points (mean $\pm$ standard deviation; lower is better). The PointNSP row reports its VQ-VAE reconstruction.

	Airplane		Chair		Car	
Method	CD (↓)	EMD (↓)	CD (↓)	EMD (↓)	CD (↓)	EMD (↓)
PointFlow	1.30 ± 0.00	5.36 ± 0.06	10.43 ± 0.02	17.54 ± 0.16	6.94 ± 0.01	12.93 ± 0.19
SoftFlow	1.19 ± 0.00	4.28 ± 0.06	11.05 ± 0.03	17.68 ± 0.08	6.82 ± 0.01	11.44 ± 0.10
ShapeGF	1.05 ± 0.00	4.42 ± 0.04	5.96 ± 0.01	12.23 ± 0.11	5.68 ± 0.01	9.26 ± 0.18
Canonical VAE	0.98 ± 0.00	3.19 ± 0.03	6.56 ± 0.02	8.60 ± 0.07	5.44 ± 0.01	6.13 ± 0.02
IP-Encoder	1.05 ± 0.00	**1.57 ± 0.00**	9.24 ± 0.00	6.19 ± 0.00	5.82 ± 0.00	**3.18 ± 0.00**
PointNSP	1.02 ± 0.01	2.71 ± 0.03	5.24 ± 0.02	6.35 ± 0.05	5.31 ± 0.02	4.92 ± 0.04
NAVE (Ours)	**0.96 ± 0.00**	2.42 ± 0.00	**4.97 ± 0.00**	**5.93 ± 0.00**	**5.12 ± 0.00**	4.67 ± 0.00

C.2 What the Ablations Control

Hierarchy and representation. Table 2(A) holds the total slot count at 21 and uses multi-directional ordering for the one-, two-, and three-scale variants. This controls slot count, but equal slot counts alone do not imply equal flattened latent dimension or equal model parameter counts. Table 2(B) separates fixed-unit comparisons ($N = 21$) from fixed-codebook comparisons ($K = 512$). These comparisons probe different capacity changes and should be interpreted separately. The $N = 336$ quantized variant improves reconstruction relative to the continuous reference while worsening generation 1-NNA, illustrating that reconstruction fidelity and ease of density modeling need not improve together.

Conditioning and ordering. Table 5(A) varies the context used for the finest scale with multi-directional ordering fixed. Part (B) varies ordering with $Z_{1:2}$ conditioning fixed. These comparisons examine the two dependencies in Eqs. 1 and 2 separately.

Noise training and refinement. Table 5(C) compares noise strategies and refinement budgets. At zero refinement steps, small-scale noise outperforms multi-scale noise; the multi-scale model improves with additional refinement. Thus the reported benefit applies to the combination of noise training and sufficient refinement. The table also exposes the resulting quality–time trade-off, rather than supporting a claim that noise augmentation alone improves generation.

C.3 Qualitative Guidance and Ordering Ablations

Figure 8 supplements the main-text ablation in Figure 6 with Airplane, Car, and Chair examples. Panel (a) compares finest-scale generation conditioned on both preceding scales, without preceding-scale guidance, and conditioned only on the second scale. Panel (b) compares multi-directional, single-directional, and bi-directional intra-scale modeling. The examples illustrate changes in global proportions and part geometry under these settings; aggregate comparisons are reported in Table 5.

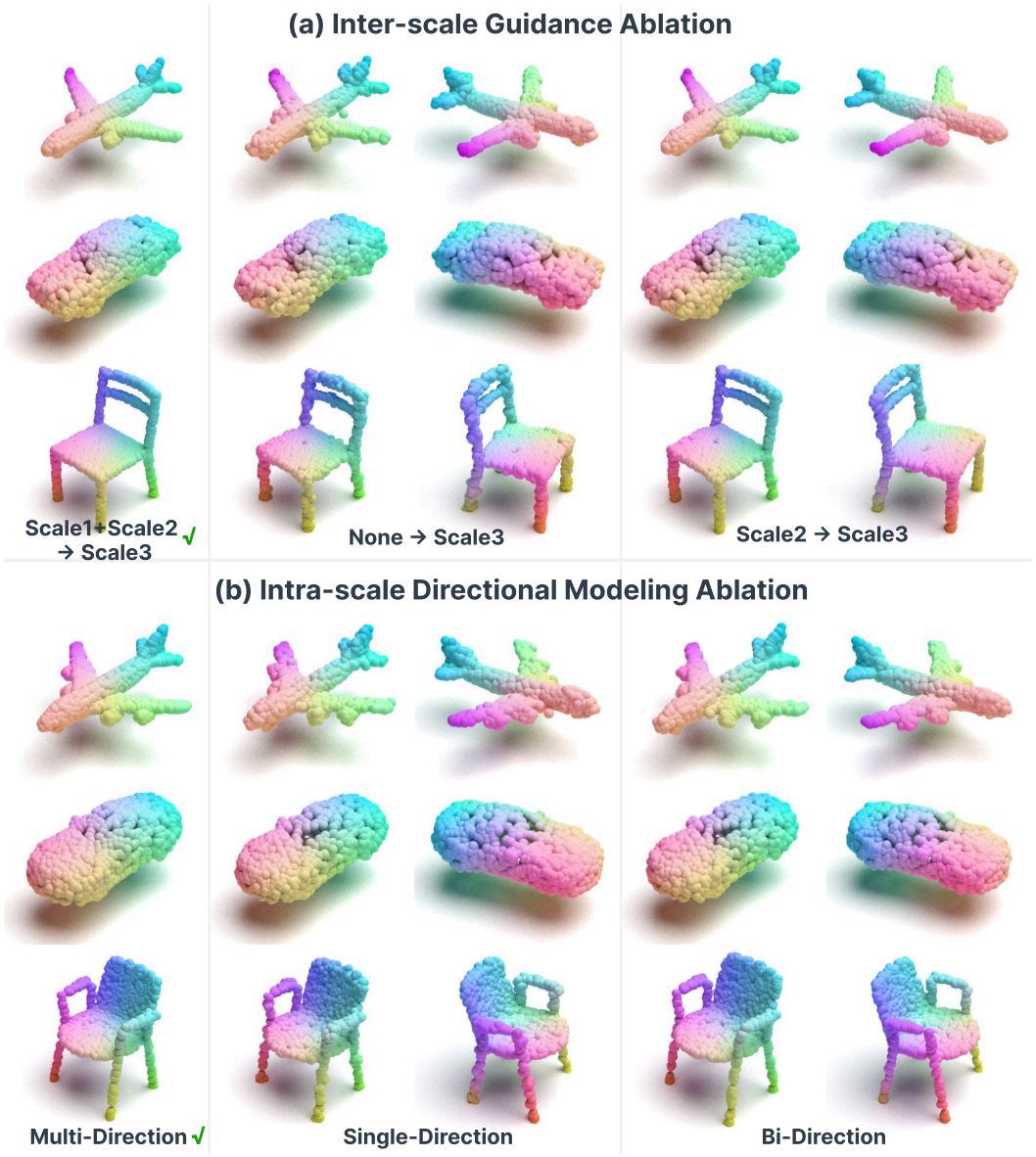

Figure 8: Additional qualitative ablations of inter-scale guidance and intra-scale directional modeling. (a) Both preceding scales ($Z_{1:2} \to Z_3$), no guidance, and second-scale-only guidance ($Z_2 \to Z_3$). (b) Multi-directional, single-directional, and bi-directional modeling. Each panel shows Airplane, Car, and Chair examples; the proposed setting is marked with a check in the left column.

D Additional Qualitative Results

The following figures supplement the main-text comparisons with progressive generation, category-specific output grids, multi-category outputs, completion and upsampling, and latent interpolation. They illustrate individual shapes and geometric transitions; the quantitative conclusions remain based on the evaluation tables.

D.1 Progressive Generation

Figure 9 displays coarse, middle, and fine outputs at 128, 512, and 2048 points. Unlike the reconstruction examples in Figure 2, these examples visualize the generation process. Each resolution gives a complete global structure, and later stages increase geometric detail.

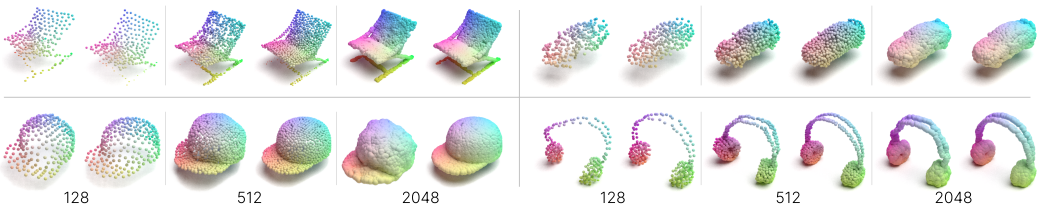

Figure 9: Progressive generation at 128, 512, and 2048 points. Each example is shown at three resolutions, with paired renderings at each resolution. The outputs illustrate complete shapes at intermediate scales.

D.2 Category-Specific Generation Examples

Figures 10–12 provide additional Airplane, Chair, and Car outputs. Each grid contains 20 examples. The grids show variations in global proportions and part structure, complementing the baseline comparisons in Figure 5. Figure 13 adds 20 Bowl examples beyond the three categories in the single-category quantitative table.

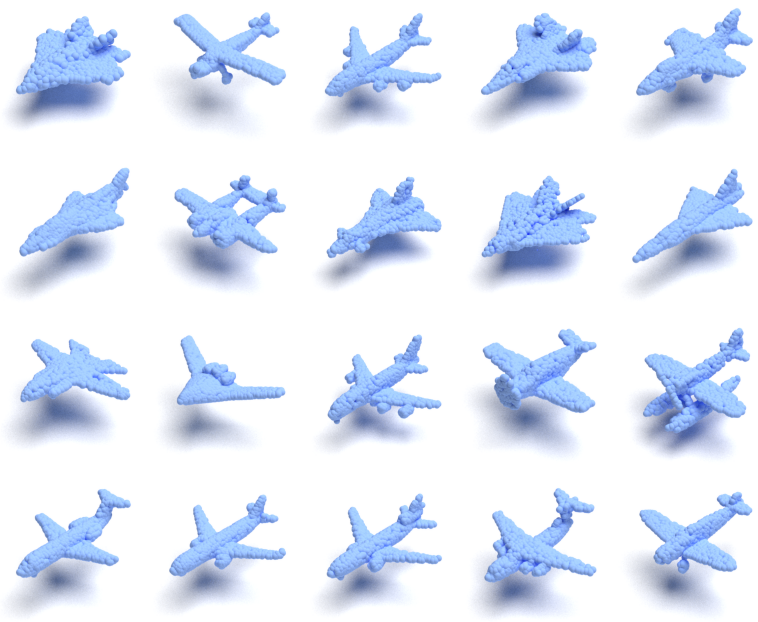

Figure 10: Additional Airplane generation examples. The grid contains 20 outputs and shows variation in wing, fuselage, and tail geometry.

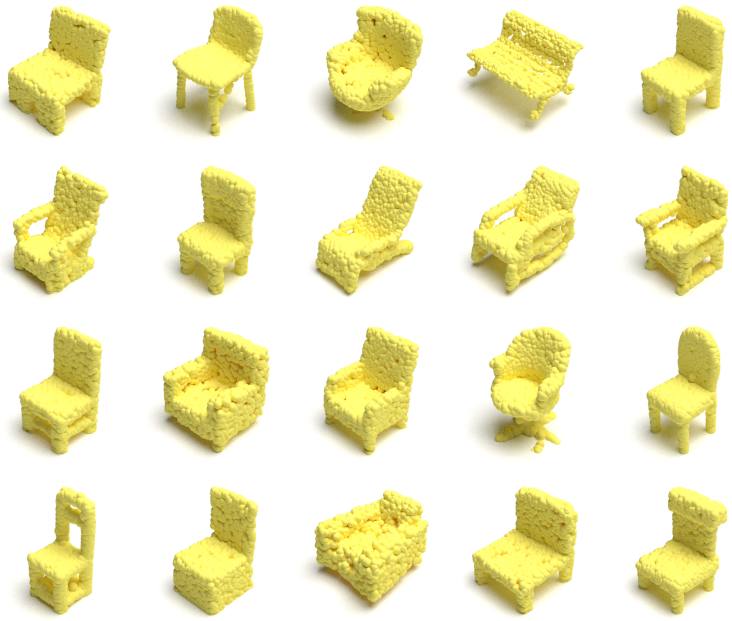

Figure 11: Additional Chair generation examples. The 20 outputs show variations in seats, backrests, armrests, and support structures.



Figure 12: Additional Car generation examples. The grid contains 20 outputs with variations in body proportions and overall shape.



Figure 13: Additional Bowl generation examples. The 20 outputs show variation in rim shape, vessel depth, and base geometry.

D.3 Multi-Category Generation

Figures 14 and 15 show two grids of multi-category generation examples, complementing Table 4. Each grid contains 50 outputs spanning multiple object types. The grids illustrate the range of generated geometry without assuming a uniform category distribution or providing a separate evaluation of all 55 categories.

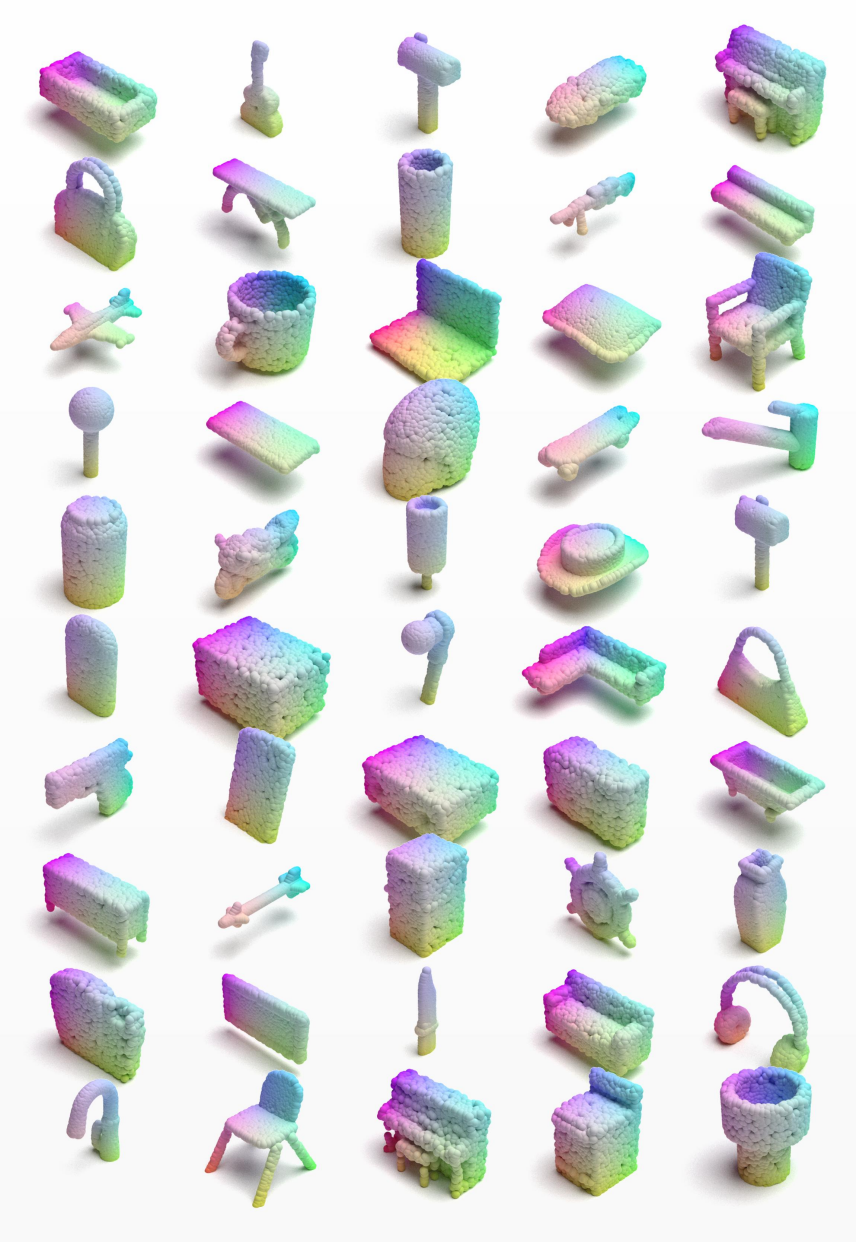

Figure 14: Additional multi-category generation examples. The grid contains 50 outputs spanning furniture, containers, vehicles, and other object types.

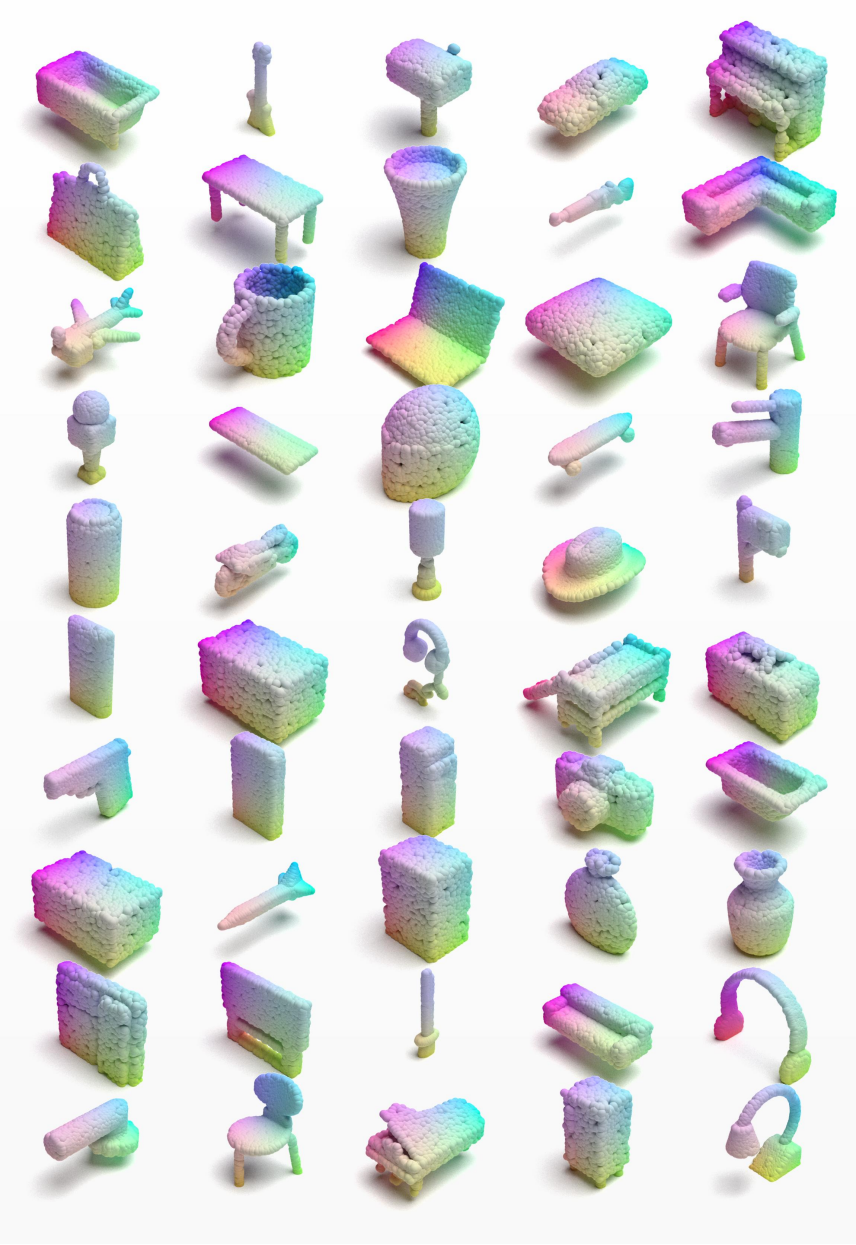

Figure 15: Further multi-category generation examples. This grid contains 50 outputs spanning multiple object types and complements Figure 14.

D.4 Completion and Upsampling

Figure 16 supplements the downstream examples in Figure 7 with Bowl and Bottle point clouds. The upper two groups show completion from partial inputs, and the lower two groups show upsampling from sparse inputs. Each group contains five input–output pairs, illustrating variations in vessel shape and local point density. These examples extend the qualitative task demonstration to additional object types; quantitative comparisons remain those reported in Table 6.



Figure 16: Additional completion and upsampling examples on Bowl and Bottle. From top to bottom, the four groups show Bowl completion, Bottle completion, Bowl upsampling, and Bottle upsampling. Within each group, inputs are shown above the corresponding outputs.

D.5 Latent Interpolation

Figures 17 and 18 supplement Figure 4 with within-category and cross-category paths decoded from interpolated NAVE latents. The within-category examples show changes in chair and airplane geometry. The cross-category examples show Table-to-Vessel and Skateboard-to-Airplane transitions, with both endpoints included. These sequences probe whether intermediate continuous representations remain decodable; they do not establish uniform probability density along the interpolation path.

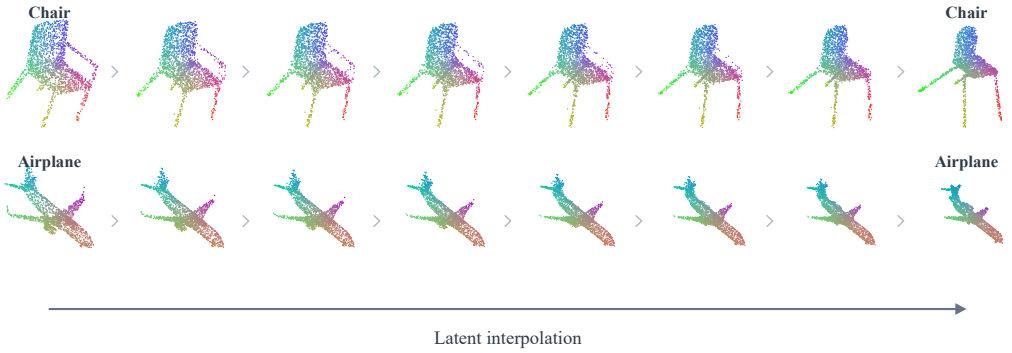

Figure 17: Additional within-category latent interpolation on Chair (top) and Airplane (bottom). Each row shows endpoint shapes and intermediate decoded shapes from left to right.

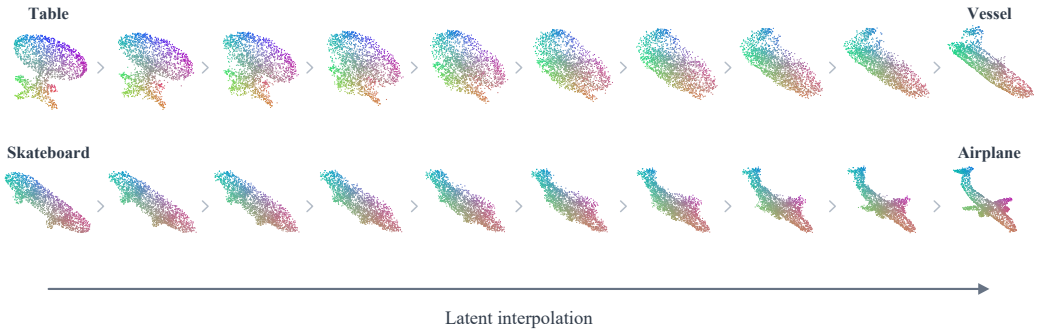

Figure 18: Additional cross-category latent interpolation from Table to Vessel (top) and from Skateboard to Airplane (bottom). Each row contains ten decoded shapes, including both endpoints.

Figure 19 provides five further interpolation sequences. The first three rows show cross-category transitions from Can to Loudspeaker, Guitar to Helmet, and Airplane to Rocket. The last two rows show within-category transitions for Airplane and Chair. Each row contains five decoded shapes, including both endpoints, and illustrates changes in overall proportions and part geometry.

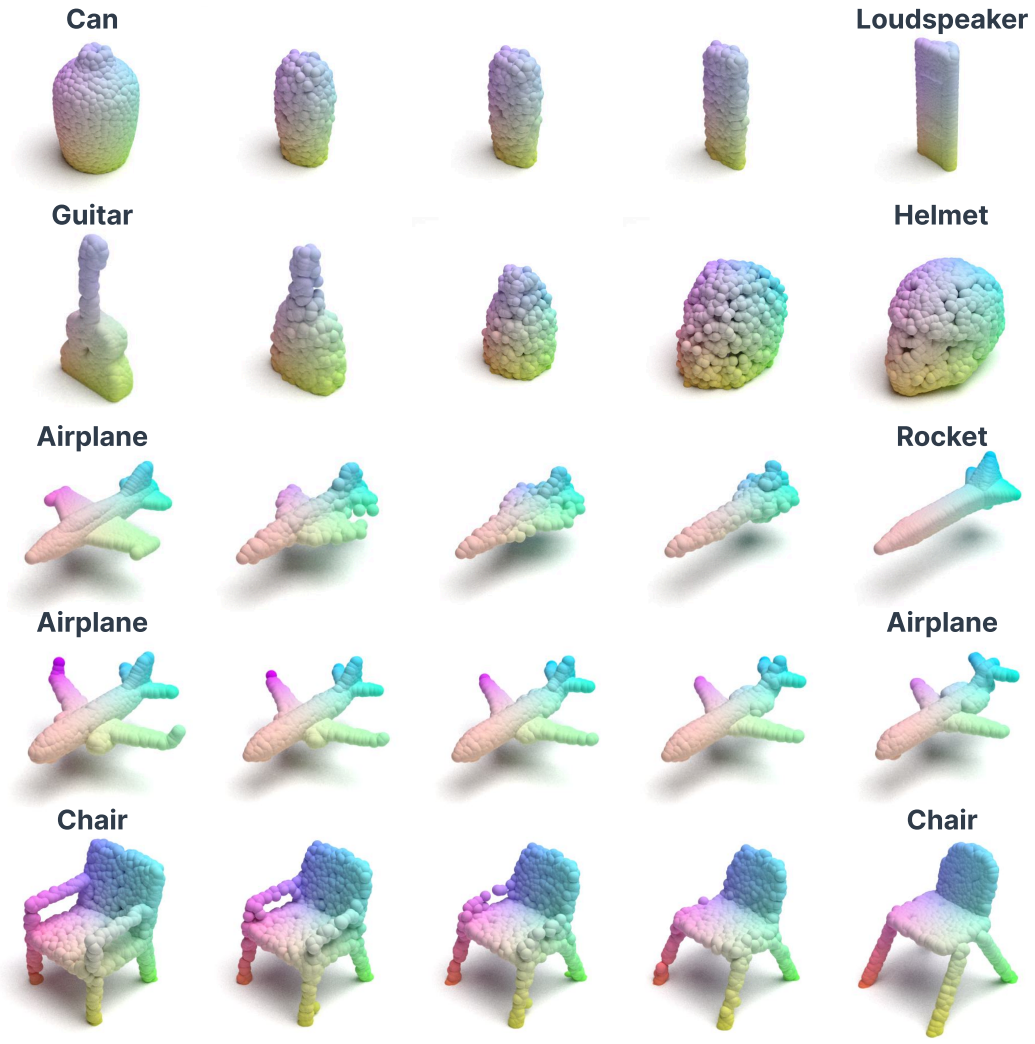

Figure 19: Further cross-category and within-category latent interpolation. From top to bottom: Can to Loudspeaker, Guitar to Helmet, Airplane to Rocket, Airplane to Airplane, and Chair to Chair. Each sequence runs from the left endpoint to the right endpoint through three intermediate decoded shapes.

E LIMITATIONS AND QUALITATIVE FAILURE MODES

The current experiments focus on object-level point clouds with a fixed finest resolution of 2048 points. They do not evaluate variable-size point sets, scene-level generation, or additional output attributes such as normals and colors. The additional generation grids also show limitations at the individual-shape level: thin parts can be irregular, part boundaries can merge, and local point distributions can be uneven. These observations motivate further analysis of fine geometry and structural validity alongside set-level generation metrics.

The generation results also vary by category and metric. For example, Airplane's 1-NNA-CD remains at 65.19 despite high coverage, and the reconstruction comparisons do not favor NAVE under every EMD evaluation. Multi-category visual examples demonstrate varied outputs but cannot replace per-category quantitative evaluation over the full training distribution. Finally, the refinement ablation shows a quality–time trade-off, so the sampling budget remains part of the method's practical operating conditions.