

# DiffFacto: Controllable Part-Based 3D Point Cloud Generation with Cross Diffusion

George Kiyohiro Nakayama<sup>1</sup> Mikaela Angelina Uy<sup>1</sup> Jiahui Huang<sup>2</sup>

Shi-Min Hu<sup>2</sup> Ke Li<sup>3</sup> Leonidas Guibas<sup>1</sup>

<sup>1</sup>Stanford University <sup>2</sup>Tsinghua University <sup>3</sup>Simon Fraser University

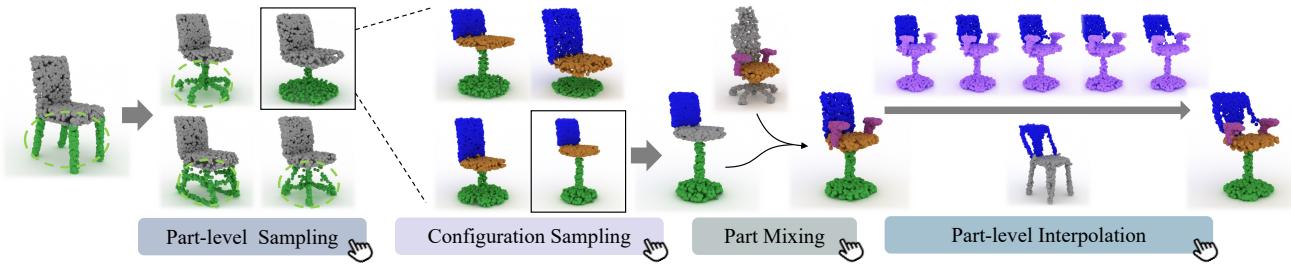


Figure 1: **DiffFacto**: Our approach tackles the task of controllable part-based point cloud generation, where we are able to generate novel shapes - novel configurations of novel parts. Our probabilistic generative model learns a *factorized* representation of shapes with a *cross diffusion* network allowing for control. We demonstrate that **DiffFacto** not only enables controllable generation but also various shape editing tasks.

## Abstract

While the community of 3D point cloud generation has witnessed a big growth in recent years, there still lacks an effective way to enable intuitive user control in the generation process, hence limiting the general utility of such methods. Since an intuitive way of decomposing a shape is through its parts, we propose to tackle the task of controllable part-based point cloud generation. We introduce **DiffFacto**, a novel probabilistic generative model that learns the distribution of shapes with part-level control. We propose a factorization that models independent part style and part configuration distributions, and present a novel cross diffusion network that enables us to generate coherent and plausible shapes under our proposed factorization. Experiments show that our method is able to generate novel shapes with multiple axes of control. It achieves state-of-the-art part-level generation quality and generates plausible and coherent shape while enabling various downstream editing applications such as shape interpolation, mixing, and transformation editing. Please visit our project webpage at <https://difffacto.github.io/>

## 1. Introduction

3D shape generation [52] is an important and popular task, where point clouds are one popular representation [53, 5, 59, 34] – due to their *simple yet powerful* expressivity

as well as data availability, *i.e.* just a set of points and can directly be acquired by sensors. However, the generation of arbitrary plausible shapes is often of limited utility, as users often have a conceptual design idea and of what they want to generate.

A prerequisite of shape generation is to be able to learn a space of all possible shapes. To represent this space, one parsimonious way is to represent them as a combination of simpler atoms, known as *parts*. In this flavor, we propose the task of *controllable part-based generation*, which aims to generate plausible novel shapes with user control over individual parts. As mentioned before, a shape is a combination of parts, thus a ‘novel shape’ can be defined in three different ways: (1) *novel configurations* of existing parts, (2) *existing configurations of novel parts*, and (3) *novel configurations of novel parts*.

The first is explored in existing graphics literature such as part retrieval [48] and shape assembly [57], while the second can be tackled by existing generation methods [53, 5, 59] trained on parts. In contrast, we tackle the third, which is the more challenging case subsuming the first and second. A challenge arises because a shape is a combination of novel parts, leading to an exponential explosion of plausible shapes while having only limited training data. A further challenge stems from enabling control as this requires an approach that can vary individual parts and configurations while still generating plausible shapes.

To this end, we introduce a new method that tackles this task in a principled way by building a *probabilistic generative model* that learns the distribution of shapes while enabling control on parts and configurations. Specifically, we propose a *factorization* that decomposes the shape space into (i) the individual canonicalized (semantic) parts, and (ii) their transformations (position and size). These factors can be sampled or encoded independently, allowing for different modes of control in generation and intuitive editing.

Our approach learns independent latent spaces for each canonicalized (semantic) part through *part stylizers*. Then conditioned on the canonicalized parts, we also introduce learn a *transformation sampler* that learns a distribution of part configurations. Naive approaches can result in mode collapse since multiple parameter configurations can output a valid shape, if conditioned only on canonicalized parts. We leverage on a sampling-based approach to learn a multi-modal distribution of part configurations through conditional Implicit Maximum Likelihood [30] (cIMLE).

To generate plausible shapes through independently sampled factors, we also introduce our *cross diffusion* network that allows for the learning of a better shape distribution under our proposed factorization. Our cross-attention diffusion network, conditions on the proposed factors, *i.e.* independent part style and transformations, in the reverse diffusion process. Our design allows each generated point in the point cloud to be informed of both the global shape as well as the local part, resulting in more plausible and coherent output shapes while still enabling control. Moreover, we also introduce a generalized forward diffusion kernel that allows the explicit encoding of each part transformations, enabling better shape reconstruction and transformation extrapolation.

We dub our method **DiffFacto**, for **f**actorized representation with cross **d**iffusion. To our best knowledge, we are the first to introduce a factorized representation that allows for control in both part styles and part configurations as we model independent part style distributions and transformation distribution, enabling each to be independently sampled. Experiments show that our approach achieves better intra-part and inter-part level scores compared to baselines. We also show that our approach generates novel and coherent shapes through a segmentation-based plausibility experiment and human study. Furthermore, we demonstrate that our approach also allows for controllable and localized shape editing on various applications such as part-level shape interpolation, shape mixing and transformation editing.

## 2. Related Work

**Point Cloud Generative Models.** The literature mainly targets at an accurate modeling of the underlying data distribution, using probabilistic tools and parameterized deep networks developed in variational auto-encoders (VAEs) [25] used by PSG [12], generative adversarial networks (GANs) [14] used by PointGAN [1], auto-regressive

models [4] used by PointGrow [46], normalizing flows [41] used by PointFlow [53] or Softflow [23], *etc.*, with proper conditioning [45, 36, 30]. Notably, the most recent success of Diffusion Models [19] for image synthesis [43] is further expanded to the domain of 3D point clouds generation [34, 60, 5, 59, 39, 21]. Among this line of works, PVD [61] and LION [59] treat the full point cloud as a *single* sample from the learned distribution in either primal or latent space. However, the points themselves from a cloud can be viewed as samples from a geometric distribution, which is explicitly modeled by, *e.g.*, DPM [34] or ShapeGF [5], and PointFlow [53] as *distribution of distributions* [53]. This enables the desirable capability to generate *arbitrary* number of points from a single shape without the need of re-training. Our work takes this approach and additionally builds a *factorized* prior to sample from allowing for part-level control, which in contrast to all previous works that only model a *single* latent space and can only generate a full shape.

**Structure-aware Shape Generation.** Different from *full-shape* generation, structure-aware methods enrich the synthesized geometry with useful structural information and allow for easy user manipulations [8]. Related literature has explored different structure representations, such as segmented parts [38, 27, 32], relationship graphs [13, 51], or layers/hierarchies [28, 16, 54], generated using either auto-regressive models [50, 39], recursive networks [28], or hierarchical models [54, 49]. Among these works, SPAGHETTI [17] uses cross-attention to mix the part latents that are decoded into an implicit shape representation. SP-GAN [31] employs a spherical proxy geometry for even finer-grained controls. Nevertheless, most of the above existing methods model the shape prior as a single latent code, and user controls are *detached* from the generative process. They hence rely on post-optimization/inversion [16] steps to maintain the shape plausibility. In contrast, our *factorized* prior allows for explicit part-based control and sampling *during* the generative process.

**Shape Editing.** In graphics, *low-level* shape editing usually involves computing and modifying geometric handles such as cages [44, 11] or skeletons [58, 3]. On the other hand, the exploitation of *high-level* semantics allow for more intuitive controls of the shape such as mixing and stitching semantic parts from a shape collection [57], tweaking the scales and deformations of the bounding boxes [33, 47], changing the status of a coarse proxy geometry [55, 35], or using natural languages [26]. They typically however start from an *existing source shape* which is edited to a desired target. In contrast, in addition to editing existing geometry, our method can also create *novel* shapes in a *controllable* manner with our introduced factorization, and enabling a wider range of applications than the prior arts.

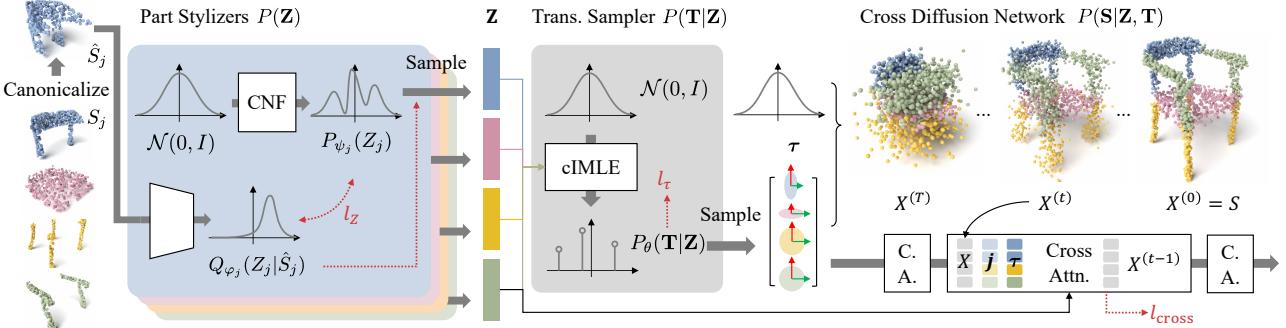


Figure 2: **Method Overview.** We factorize the 3D shape distribution into three key components, containing  $m$  **part stylizers** for each part that model the shape prior  $P(\mathbf{Z})$ , a **transformation sampler** that models the conditional distribution of transformations given the part latents  $P(\mathbf{T}|\mathbf{Z})$ , and a **cross diffusion network** that samples the point cloud jointly considering the part geometry and their configurations  $P(\mathbf{S}|\mathbf{Z}, \mathbf{T})$ . Red dashed lines indicate losses incorporated in the training stage. For definitions of variables, see Sec. 3.

### 3. Problem Overview

Our goal is to learn a *controllable* generative model on 3D shapes. Given a set of shapes  $\mathbf{S} = \{S^{(i)}\}$ , we want to learn a distribution  $P(\mathbf{S})$  from which we can sample with part-level control. To generate realistic and plausible shapes, a *shape prior* is commonly learned. Existing works [53, 5, 34, 18] model the shape prior as one random variable  $w \sim P(W)$ , *i.e.* a single global latent code, which *does not* allow for any level of structure-aware control on the shape being sampled/generated.

We introduce a **factorized** representation of the prior in order to obtain more localized control of the generation process. An intuitive granularity for users to control shape generation is through a natural decomposition of shapes into semantic parts [37], which we leverage on in our proposed factorization. For the rest of the paper, we assume that shapes in  $\mathbf{S}$  from a given category have a predefined set of  $m$  semantic parts.

A shape  $S^{(i)}$  is decomposed into its semantic parts  $\{S_j^{(i)}\}_{j=1,\dots,m}$ , where  $S_j^{(i)}$  is the geometry of each shape part the superscript is omitted in what follows for brevity. We further factorize  $S_j$  into its *canonicalized geometry*,  $\hat{S}_j$ , *i.e.* **part style**, and its corresponding **instancing transformation**,  $T_j \in \mathbb{R}^{D_1}$ . We then model independent distributions  $P(Z_j)$  for each canonicalized part, where  $Z_j \in \mathbb{R}^{D_2}$ ,  $j = 1, \dots, m$  is the canonicalized part latents, which we call as **part stylizers**. For simplicity, we use bold type for random variables and samples on the collection for all parts, e.g.  $\mathbf{Z} = \{Z_j\}$ , uppercase letters are random variables and lowercase letters are their corresponding samples, e.g.  $z_j \sim Z_j$ ,  $\tau_j \sim T_j$ . Conditioned on all the part style latents  $\mathbf{Z}$ , we further model a distribution on their transformations,  $P(\mathbf{T}|\mathbf{Z})$ , where  $\mathbf{T} = \{T_j\}$ . The shape prior distribution is then the *joint* distribution of individual part styles and their transformations. Our proposed shape factorization is then given as:

$$P(\mathbf{S}) = \int \int P(\mathbf{S}|\mathbf{z}, \boldsymbol{\tau}) P(\boldsymbol{\tau}|\mathbf{z}) P(\mathbf{z}) d\mathbf{z} d\boldsymbol{\tau}, \quad (1)$$

where our shape prior is now *factorized* as  $P(\mathbf{Z}, \mathbf{T}) = \prod_{j=1}^m P(Z_j)P(\mathbf{T}|Z_j)$ . Our factorization allows for control in our learned generative model, since we are able to independently sample from each part style latent distribution  $z_j \sim P(Z_j)$  and a valid set of transformations  $\boldsymbol{\tau} \sim P(\mathbf{T}|Z_j)$ , thus introducing multiple different knobs for variation. We detail our joint prior  $P(\mathbf{Z}, \mathbf{T})$  in § 4.2. To capture and generate more plausible and holistic shapes, we further introduce our *cross diffusion network* that models  $P(\mathbf{S}|\mathbf{Z}, \mathbf{T})$ , a conditional distribution of points on the shape’s surface. We elaborate our diffusion-based network and also introduce a generalized forward kernel in § 4.3. We showcase that our proposed factorization not only allows us to sample and generate diverse and plausible shapes, but also enables multiple shape editing and variation applications in § 4.4.

### 4. Method

We learn a shape distribution  $P(\mathbf{S})$  given segmented shapes, represented as a point cloud  $S = \{x_k\}_{k=1,\dots,N}$ ,  $x_k \in \mathbb{R}^3$  with their semantic parts  $\{S_j\}_{j=1,\dots,m}$ , where  $S = \cup_{j=1}^m S_j$ . Each part  $S_j$  is decomposed into its part style  $\hat{S}_j$  and transformation  $\tau_j$ . Concretely, the part style  $\hat{S}_j$  is the canonicalized part geometry given by the transformation  $\tau_j \in \mathbb{R}^6$  representing the shift ( $c_j \in \mathbb{R}^3$ ) and axis-aligned scale ( $s_j \in \mathbb{R}_{>0}^3$ ), giving us:

$$S_j = \text{Diag}(s_j)\hat{S}_j + c_j.$$

Our method models the data distribution  $P(\mathbf{S})$  with a factorized joint prior  $P(\mathbf{Z}, \mathbf{T})$  and has three components: (i) independent **part stylizers** that model  $P(Z_j)$  for  $j = 1, \dots, m$ , representing part styles, (ii) a **transformation sampler** that models a distribution  $P(\mathbf{T}|\mathbf{Z})$  of the part transformations

conditioned on their styles, and (iii) a **cross diffusion network** that models the conditional distribution  $P(S|\mathbf{Z}, \mathbf{T})$  of points on the surface of the shape given the factorized prior. Fig. 2 illustrates our method overview.

## 4.1. Training Objective

We first derive the training objective for our factorized joint prior  $P(\mathbf{Z}, \mathbf{T})$ . We maximize the likelihood of our learned distribution  $P(\mathbf{S})$  through the evidence lower bound. We let  $\psi, \theta, \phi$  be the model parameters of our part stylizer, transformation sampler and cross-diffusion network, respectively. In addition, we assume an evidence distribution  $Q_\varphi(\mathbf{Z}, \mathbf{T}|S)$  that can be formulated as:

$$\begin{aligned} Q_\varphi(\mathbf{Z}, \mathbf{T}|S) &= Q_\varphi(\mathbf{Z}|S) Q(\mathbf{T}|S) \\ &= Q(\mathbf{T}|S) \prod_{j=1}^m Q_{\varphi_j}(Z_j|S_j) \\ &= \prod_{j=1}^m Q_{\varphi_j}(Z_j|\hat{S}_j), \end{aligned} \quad (2)$$

where  $Q(\mathbf{T}|S)$  is deterministic since we assumed known segmentation  $S = \{S_j\}$ , and we assume  $Z_{j_1} \perp Z_{j_2} \forall j_1, j_2 \in \{1, \dots, m\}, j_1 \neq j_2$ , i.e.  $\mathbf{Z}$  and  $\mathbf{T}$  are conditionally independent given  $S$ . Thus, the evidence lower bound (ELBO) for the likelihood can be derived as:

$$\begin{aligned} &\mathbb{E}_S [\log P_{\phi, \psi, \theta}(S)] \\ &= \mathbb{E}_S \left[ \log \iint P_\phi(S|\mathbf{z}, \boldsymbol{\tau}) P_{\psi, \theta}(\mathbf{z}, \boldsymbol{\tau}) d\mathbf{z} d\boldsymbol{\tau} \right] \\ &= \mathbb{E}_S \left[ \log \iint \frac{P_\phi(S|\mathbf{z}, \boldsymbol{\tau}) P_\psi(\mathbf{z}) P_\theta(\boldsymbol{\tau}|\mathbf{z})}{Q_\varphi(\mathbf{z}, \boldsymbol{\tau}|S)} Q_\varphi(\mathbf{z}, \boldsymbol{\tau}|S) d\mathbf{z} d\boldsymbol{\tau} \right] \\ &\geq \mathbb{E}_{S, \mathbf{z}} \left[ \underbrace{\log P_\phi(S|\mathbf{z}, \boldsymbol{\tau})}_{\mathcal{L}_{\text{recon}}} + \underbrace{\sum_{j=1}^m \log \frac{P_{\psi_j}(z_j)}{Q_{\varphi_j}(z_j|\hat{S}_j)}}_{\mathcal{L}_Z} + \underbrace{\log P_\theta(\boldsymbol{\tau}|\mathbf{z})}_{\mathcal{L}_{\boldsymbol{\tau}}} \right], \end{aligned} \quad (3)$$

where the inequality is through Jensen's (see supplement). We now go through each of our components, and how they are trained to maximize this derived training objective with the corresponding loss functions (denoted as  $l_{(\cdot)}$ ).

## 4.2. Factorized Joint Prior $P(\mathbf{Z}, \mathbf{T})$

Our factorization allows us to define multiple random variables to control the generative model and output plausible shapes. These random variables can be independently sampled and are the ‘control knobs’ in the generation process, thus allowing for user-controllability. These random variables are  $Z_j$  to control per-part style through our part stylizers and  $\mathbf{T}$  that controls the set of part transformation through our transformation sampler. Moreover, this also allows the network to consolidate part information, e.g. the canonicalized geometry of a rectangular back of the bench is the same as a dining chair when transformation is factored out. This decomposition also enables resampling certain part geometries  $Z_j$  while keeping the rest fixed, creating

variations in part configurations by resampling  $\mathbf{T}$ , and local shape editing through encoding and modifying  $z_j$  or  $\boldsymbol{\tau}_j$ .

**Part Stylizer.** The part stylizer learns to model the independent distributions  $P(\hat{S}_j)$  of part styles for  $j = 1, \dots, m$ . Each  $\hat{S}_j$  is encoded into a part latent code  $Z_j$  with encoder  $Q_{\varphi_j}(Z_j|\hat{S}_j)$ , representing the evidence distribution. We use a continuous normalizing flow [9] (CNF) model to learn part priors  $P_{\psi_j}(Z_j)$  with parameters  $\psi_j$ . Formally, we have the part stylizer loss  $\ell_Z$  given by:

$$\begin{aligned} \ell_Z &= \sum_{j=1}^m \text{KL} \left( Q_{\varphi_j} \left( Z_j | \hat{S}_j \right) \| P_{\psi_j} (Z_j) \right) \\ &= - \sum_{j=1}^m \mathbb{E}_{z_j \sim Q} [\log P_{\psi_j}(z_j)] + H \left( Q_{\varphi_j} \left( Z_j | \hat{S}_j \right) \right) \end{aligned} \quad (4)$$

where  $H$  is the entropy and  $P_{\psi_j}(Z_j)$  is a complex distribution transformed from Gaussian. This is equivalent to maximizing  $\mathcal{L}_Z$  in Eq (3). See supplement for more details on CNF.

**Transformation Sampler.** Given part styles  $\mathbf{z}$ , there are different plausible configurations, i.e. multiple sets of transformations, that result in valid shapes. Hence, we cannot simply regress the transformations  $\mathbf{T}$  for given part styles  $\mathbf{z}$ , leading us to model a conditional *distribution* of transformations  $P(\mathbf{T}|\mathbf{Z})$ . Learning this distribution is non-trivial because of three main reasons: (i) we have to learn a diverse set of shape variations captured only by the transformation parameters, e.g. a dining chair and a bench can have the same set of part styles, (ii) each training example  $S \in \mathbf{S}$  only provides one-to-one pairs  $(\mathbf{z}, \boldsymbol{\tau})$  of part styles and transformations, and (iii) the desired conditional distribution  $P(\mathbf{T}|\mathbf{Z})$  may be *multimodal*.

To satisfy these properties, we leverage on conditional Implicit Maximum Likelihood (cIMLE) [30] that trains an implicit generative model,  $P_\theta(\mathbf{T}|\mathbf{Z})$  in our case, by encouraging *some* generated output to match the observation from  $S$ , in contrast to unimodal approaches [36, 45] that enforces *all* generated outputs to match the observation leading to mode collapse. Concretely, transformation sampler  $T_\theta$  outputs samples  $\boldsymbol{\tau}_k = T_\theta(\mathbf{z}, y_k)$  for part style latents  $\mathbf{z}$  and random latent variable  $y_k \sim \mathcal{N}(0, I)$ ,  $y \in \mathbb{R}^{D_\tau}$ . We sample multiple latents  $y_1, \dots, y_K$  and encourage that at least one of them matches the observed data  $S$ . As shown in IMLE [29], maximizing the likelihood is then equivalent to minimizing the loss:

$$\ell_{\boldsymbol{\tau}} = \sum_{S \in \mathbf{S}} \min_{k=1, \dots, K} \ell_{\text{fit}}(T_\theta(\mathbf{z}_S, y_k), \boldsymbol{\tau}_S), \quad (5)$$

where  $\boldsymbol{\tau}_S$  is the observed part transformations for shape  $S$ . We define  $\ell_{\text{fit}}$  as the distance between the generated set  $\{\boldsymbol{\tau}_k\}$  and observation  $\boldsymbol{\tau}_S$  summed across all parts, given as

$$\ell_{\text{fit}}(\boldsymbol{\tau}, \boldsymbol{\tau}_S) = \sum_{j=1}^m \|c_j - c_{S,j}\|_2^2 + \|\log s_j - \log s_{S,j}\|_2^2.$$

Minimizing  $\ell_{\tau}$  is equivalent to maximizing  $\mathcal{L}_{\tau}$  in Eq (3).

### 4.3. Cross Diffusion Network

In order to capture plausible and holistic shapes given our proposed factorization, we model the conditional shape distribution  $P(\mathbf{S}|\mathbf{Z}, \mathbf{T})$  given part style latent codes and part transformations with our *cross diffusion network*. Specifically, we represent a shape as a distribution of points on its surface. Given segmented shapes  $S$  with  $M$  parts, we model  $P(x|\mathbf{Z}, \mathbf{T}, j) \forall x \in \mathbb{R}^3$  as the probability that  $x$  lies on the surface of part  $S_j$ . Since we use part semantic labels as an additional condition, this also allows us to output segmented shapes by specifying  $j$ . Each point is treated as an independent sample (denoted by the random variable  $X$ ), which leads to the conditional likelihood of a shape  $S$ :

$$P(S|\mathbf{Z}, \mathbf{T}) = \prod_{j=1}^m \prod_{x \in S_j} P(x|\mathbf{z}, \boldsymbol{\tau}, j)^1. \quad (6)$$

Our cross diffusion network leverages on denoising diffusion probabilistic model (DDPM) to learn the conditional likelihood  $P_{\phi}(\mathbf{Z}, \mathbf{T}, j)$  through an iterative denoising process. DDPM models a probability distribution using a reverse process which is a Markov chain with a fixed prior, and to learn DDPM, we approximately maximize the likelihood with the forward process as the approximate posterior. Instead of directly maximizing each conditional likelihood [34, 19], we reparameterize learn  $\varepsilon_{\phi}$  that predicts the noise at a timestep  $t$  given the noisy data point  $x^{(t)}$  and minimize the distance between the predicted noise  $\varepsilon_{\phi}$  and the ground truth noise  $\varepsilon$ . Thus maximizing  $\mathcal{L}_{\text{recon}}$  in Eq (3) is equivalent to minimizing

$$\ell_{\text{cross}} = \sum_{j=1}^m \sum_{x \in S_j} \mathbb{E}_{\varepsilon, \mathbf{z}, t}^2 \left[ \left\| \varepsilon - \varepsilon_{\phi}(x^{(t)}, \mathbf{z}, \boldsymbol{\tau}_S, j, t) \right\|_2^2 \right], \quad (7)$$

where the predicted noise  $\varepsilon_{\phi}$  is conditioned on part styles  $\mathbf{z}$ , transformations  $\boldsymbol{\tau}$ , semantic label  $j$  and timestep  $t$ . Our cross diffusion network uses our introduced *generalized* forward kernel that allows for the preservation of information from part transformations ( $\tau_j = (c_j, s_j)$ ) in the forward diffusion process while only adding noise to the canonicalized part geometries, *i.e.* part styles. We also show the reverse process of our generalized forward kernel can also be similarly reparameterized arriving at the same loss in Eq (7). Fig. 2 shows one example for the forward diffusion process. Our experiments show that this modification allows for better reconstruction quality and part transformation extrapolation.

We use a cross-attention network with  $L$  cross attention layers to instantiate  $\varepsilon_{\phi}$ . For a timestep  $t$ , the network predicts  $x^{(t-1)}$  conditioned on  $(x^{(t)}, j, \tau_j)$ . The input to each cross attention layer attends to  $m$  tokens each being the

<sup>1</sup>By the same assumption of a deterministic mapping between  $x$  and  $j$ .

<sup>2</sup> $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ,  $\mathbf{z} \sim Q_{\varphi}(\mathbf{Z}|\hat{S})$ , and  $t \sim \text{Uniform}\{1, \dots, T\}$

concatenation of  $(z_j, \tau_j, j, t)$ , for  $j = 1, \dots, m$ . This design allows a point  $x^{(t)}$  to be informed of both the global shape through the  $m$  tokens and the local part by concatenating the coordinate  $x^{(t)}$  with its corresponding part label and transformation, enabling us to capture and generate more plausible and holistic shapes.

**Generalized Forward Kernel.** Finally, we introduce a generalization of the forward kernel used to add noise  $\varepsilon$  to the data points  $x$ . Existing works [34, 60] use a forward kernel that diffuses all points on the surface to the standard unit Gaussian. We show that our generalized forward kernel is theoretically equivalent to diffusing all points to a *scaled* and *shifted* Gaussian. Our modification allows to incorporate an additional prior (scale and shift) to the forward process. Specifically, for a  $d$  dimensional diffusion process, a shift  $\mu \in \mathbb{R}^d$  and variance  $\Sigma \in \mathbb{R}^{d \times d}$  parameter is incorporated into the forward kernel so that it becomes:

$$\begin{aligned} & Q(X^{(t)}|x^{(t-1)}, \mu, \Sigma) \\ &= \mathcal{N}\left(\sqrt{\alpha_t}x^{(t-1)} + (1 - \sqrt{\alpha_t})\mu, (1 - \alpha_t)\Sigma\right), \end{aligned} \quad (8)$$

for  $t = 1, \dots, T$ . Here  $\alpha_t$ 's are variance schedule hyperparameters. As  $T \rightarrow \infty$ , we show that the final distribution approaches a parameterized Gaussian with mean  $\mu$  and variance  $\Sigma$  (see supplement for the full proof):

$$\begin{aligned} & Q(X^{(T)}|x^{(0)}, \mu, \Sigma) = \mathcal{N}\left(\sqrt{\bar{\alpha}_T}x^{(0)} + (1 - \sqrt{\bar{\alpha}_T})\mu, (1 - \bar{\alpha}_T)\Sigma\right) \\ &+ (1 - \sqrt{\bar{\alpha}_T})\mu, (1 - \bar{\alpha}_T)\Sigma \cong \mathcal{N}(\mu, \Sigma). \end{aligned} \quad (9)$$

Here  $\bar{\alpha}_T = \prod_{t=1}^T \alpha_t$ . Note that the standard forward kernel is a special case of our generalization by setting  $\mu = \mathbf{0}$  and  $\Sigma = \text{diag}(\mathbf{1})$ . For our task, we set  $\mu = c_j$  and  $\Sigma = \text{Diag}(s_j^2)$  for points  $x \in S_j$ .

Please see supplement for more details and full derivations. The total loss is then  $\ell_{\text{total}} = \ell_{\text{recon}} + \lambda_1 \ell_{\mathbf{Z}} + \lambda_2 \ell_{\boldsymbol{\tau}}$ .

### 4.4. Enabling Shape Editing

Our method does not only allow for shape generation through *sampling* from individual part style  $P(Z_j)$  and transformation  $P(\mathbf{T}|\mathbf{Z})$  distributions, but it is also able to *encode* specified parts and *modify* them, allowing for local shape editing and controllable variation synthesis. We highlight that enabling local edits/variations is non-trivial as there is a tension between *preservation* and *adaptation* post-edit. In other words, an ideal edit would keep the unmodified parts unchanged as much as possible, but at the same time still adapt the changes required to maintain a plausible shape after incorporating the specified edit.

1) In the simplest setting, our framework can be trained as an *autoencoder* by deterministically regressing each part

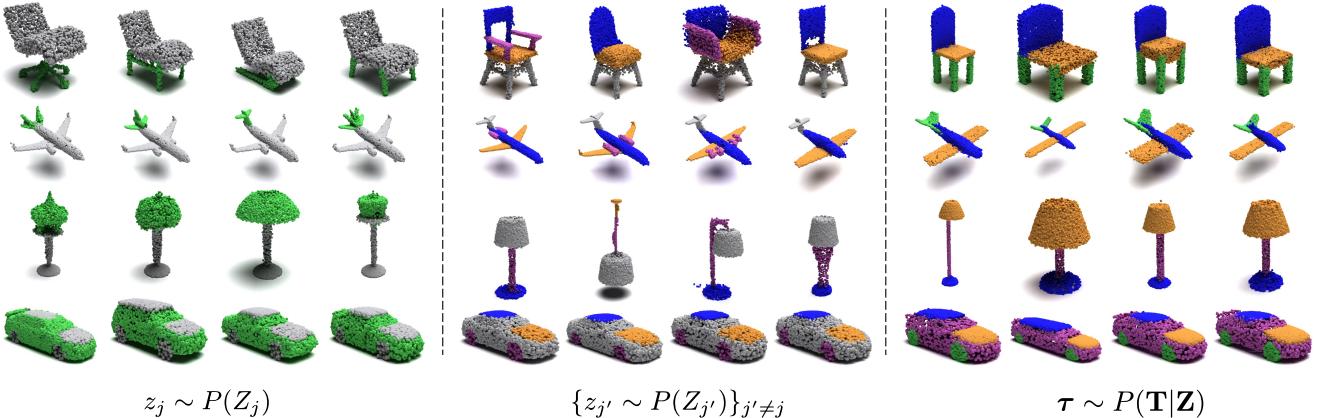


Figure 3: **Generated Shapes with Controlled Variation.** (Left) Re-sampling a selected part style while keeping the rest fixed. (Middle) Fixing a selected part while re-sampling the rest. (Right) Generating multiple part configurations for a given set of part styles. Gray refers to the fixed part while colored parts are being modified.

latent  $Z_j$  and directly feeding the part transformations  $\tau_S$  for a shape  $S$ . The training objective is simply equivalent to  $\ell_{\text{recon}}$ . 2) A step further from the autoencoder set-up is to *synthesize variations* of a shape by training the transformation sampler. Given  $S$ , we deterministically encode the part styles, and then either a) sample different transformations  $\tau$ , or b) manually edit selected part transformations  $\tau_j$  from  $\tau_S$ , achieving variations on (part) transformation configurations while keeping part styles fixed. This set-up also allows for shape mixing-and-matching with transformation variations. 3) Moreover, we also enable *local* editing where specified part  $j$  of a shape can change while keeping the rest fixed that is done by either a) resampling the corresponding part latent code from  $P(Z_j)$  or b) interpolating between part styles. The applications are shown in the results section.

## 5. Results

### 5.1. Dataset and Evaluation Protocol

We use four classes from ShapeNet [6] dataset: chair, airplane, lamps, and cars. We train/test the networks per object class with the splits provided by [56]. Each category contains 3053, 2349, 1261, 740 training shapes and 704, 341, 286, 158 test shapes, respectively. The semantic labels for all classes come from [56]. See supplement for implementation details — network architecture, training time, etc.

For our task on controllable part-based generation, we propose to measure intra-part and inter-part level scores. For more details, please refer to the supplement.

**Intra-part Score.** We evaluate the quality of individual part distributions  $P(S_j) \forall j$  using the standard generation metrics following [53]: minimum matching distance (MMD-P), coverage (COV-P) and 1-NN classifier accuracy<sup>3</sup>(1NNA-P), measuring the similarity between the distributions of canonicalized parts of the generated shapes compared to the test set from [56] of segmented shapes, where the score is

Chair	MMD-P ( $\downarrow$ )	COV-P ( $\uparrow$ )	1NNA-P
PointFlow [53]	4.68	27.3	87.77
DPM [34]	4.17	28.2	85.65
ShapeGF [5]	3.52	42.3	68.65
LION [59]	3.99	35.1	69.25
<b>DiffFacto (Ours)</b>	<b>3.27</b>	<b>42.5</b>	<b>65.23</b>

Airplane	MMD-P ( $\downarrow$ )	COV-P ( $\uparrow$ )	1NNA-P
PointFlow [53]	4.61	32.0	86.11
DPM [34]	3.52	37.7	78.74
ShapeGF [5]	3.50	40.0	72.04
LION [59]	3.68	38.8	68.73
<b>DiffFacto (Ours)</b>	<b>3.20</b>	<b>46.2</b>	<b>68.72</b>

Table 1: **Global Shape Code Baselines.** MMD-P score is multiplied by  $10^{-2}$ . COV-P and 1NNA-P are reported in %.

computed for each part and averaged across all Parts (hence the suffix ‘-P’).

**Inter-part Score.** We also measure the local part-to-part coherence of the generated shape as having control in individual parts may result in disjoint or incoherent outputs. We use a snapping metric (SNAP) that measures local connectivity between independently generated parts, given as the Chamfer distance (CD) between the  $N_{\text{snap}}$ -closest points between connected parts. We report the average score across all connections of all generated samples.

**Plausibility.** We also further evaluate the plausibility of our generated shapes by using them to augment training datasets for point cloud segmentation. The idea is if our approach generates *novel* and coherent shapes with part labels then using them for data augmentation would improve the segmentation score of part segmentation networks.

<sup>3</sup>Arrow for 1NNA-P is left out in the table because an optimal score for this metric is 50 %.

	Ctrl-ShapeGF	Ctrl-LION	<b>DiffFacto</b> (Ours)
SNAP ( $\downarrow$ )	41.12	31.76	<b>13.32</b>

Table 2: **Control-Enabled Baselines.** Snapping metric on three connections for chairs: back to leg or seat; seat to legs; and arms to seat or back. ( $CD \times 10^{-2}$ ).

**Human Study.** We also conduct a user study to evaluate controllability, methods being evaluated each generate a triplet of edited shapes, and users are asked to select the triplet containing shapes that are most plausible, where an ‘abstain’ option can be selected. An edit for a given shape is defined as resampling a pre-selected parts to output a novel shape where plausibility is measured.

## 5.2. Baselines

**Global Shape Code.** Existing point cloud generation works do not explicitly model individual parts as they sample from a single *global* shape distribution, which *do not* provide individual part-level control. To evaluate individual part distributions and measure the intra-part score, we use a pre-trained part segmenter [40] to decompose generated shapes into parts. We compare against recent works PointFlow [53], DPM [34], ShapeGF [5] and LION [59] on intra-part level scores. Inter-part scores are not measured as these works directly generate a full shape.

**Control-Enabled.** We also introduce new baselines that allow for part-level control. We modify ShapeGF [5] and LION [59] to have part-level control (prefixed by “Ctrl-”) by modeling independent part distributions, *i.e.*  $P(\mathbf{S}) = \prod_j P(S_j|w_j)$ , where each part latent distribution is modeled with a (hierarchical) variational encoder  $Q(w_j|S_j)$ . These baselines allow for independent sampling at the part-level unlike existing approaches. We measure the inter-part score for these baselines to evaluate the coherence of the generated shape. Intra-part scores are not measured as these baselines are individually trained per part.

## 5.3. Controllable Shape Generation

Fig. 3 shows shapes generated by sampling from different components in our factorization enables both control in part style and part configurations: Our approach is able to *sample* (left) or *fix* (middle) a specified part, and we are also able to generate various plausible configurations of the shape (right) with fixed part styles.

**Comparison with Global Shape Code Baselines.** Tab. 1 shows the intra-part scores of our approach compared to the global shape baselines, showcasing that our approach learns better individual part-level distributions.

**Comparison with Control-Enabled Baselines.** Tab. 2 shows the inter-part scores of our approach compared to the control-enabled baselines. We output more coherent shapes

	Orig.	+ Multi (700)	+ Control (60)
PointNet	0.709	<b>0.788</b>	<u>0.780</u>
PointNet++	0.800	<b>0.808</b>	<u>0.801</u>

Table 3: **Plausibility score.** mIOU on the cars from [56] trained original and augmented datasets.

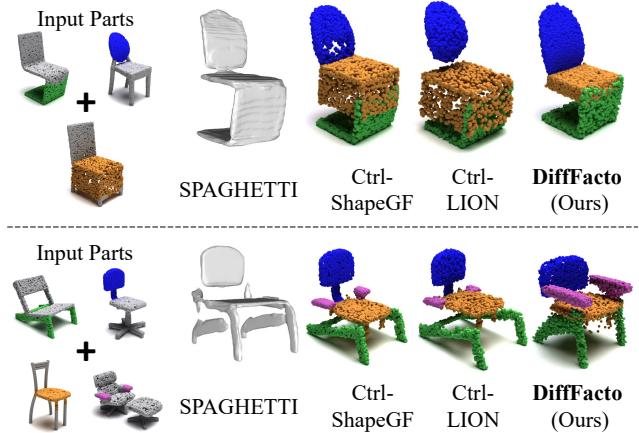


Figure 4: **Part Style Mixing.** The colored parts from the left are selected and mixed to provide the shapes on the right.

than the baselines that naively enable part-level control without the modeling of part relationships

**Plausibility.** We use the car category containing the least training data originally with 704 shapes (Orig.), and augment it with 700 randomly generated shapes by DiffFacto (+ Multi). Moreover, we test our capability for control by augmenting with only 60 (controlled) race cars, with very few examples in the original training data. Tab. 3 shows that in both cases off-the-shelf part segmentation networks improve by augmenting the training set with our generated shapes.

**Human Study.** Our human study has 100 participants comparing our approach with the control-enabled baselines, *i.e.* Ctrl-ShapeGF and Ctrl-LION. We drew 10 shapes from each of the methods with randomly selected parts to edit, and on average the participants favour **85%** (8.5 out of 10) of our generated shapes more than other baselines.

## 5.4. Shape Editing

We demonstrate that our controllable generation approach also allows for various shape-editing applications.

**Part Style Mixing.** We showcase our ability for part style mixing in Fig. 4. We compare with SPAGHETTI [18], an implicit-based shape editing work, as well as our control-enabled baselines Ctrl-ShapeGF and Ctrl-LION<sup>4</sup>. As shown, when selecting a combination of parts from different shapes, our approach generates a novel and coherent shape from

<sup>4</sup>We note that SPAGHETTI [18] requires mesh supervision.

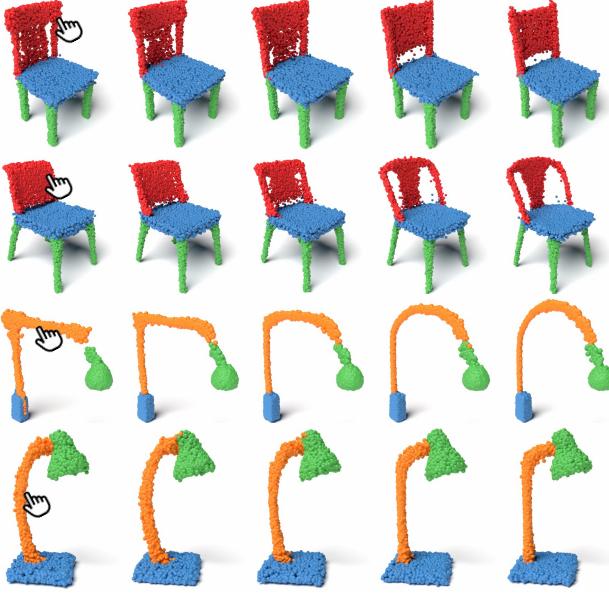


Figure 5: **Part Interpolation.** We interpolate the chair backs (red) in the first 2 rows and the lamp poles (orange) in the last 2 rows (indicated by the hand icons).

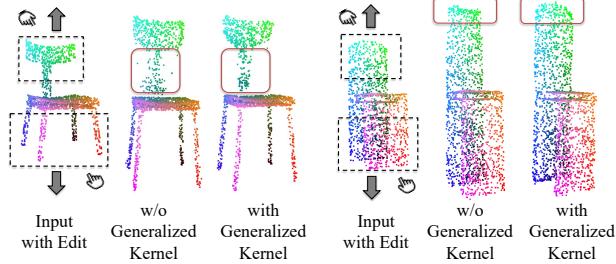


Figure 6: **Part Editing.** In both examples the user stretches the lengths of the chair backs and legs by modifying the corresponding transformations.

different input parts, compared to other approaches that are unable to adapt the parts to produce a plausible output.

**Part-level Interpolation.** Fig. 5 qualitatively shows our part-level interpolation performance, where for each example we interpolate only one selected part latent  $z_j$ . Thanks to our factorized probabilistic formulation, we are able to interpolate only the selected part while keeping the geometry of the other parts unchanged. In the meantime, the *transformations* of the other parts are automatically adapted to make the shape globally coherent.

**Transformation Editing.** Our approach also enables direct user editing on the part transformations  $\tau_j$  for a selected part  $S_j$ . We directly optimize  $y$  to find  $\tau$  that satisfies the edit while still traversing along the space of valid part configurations. As Fig. 6 (left) shows, elongating the chair back retains its thin geometric structure while still keeping the shape plausibility.

	Separate	Post Transform	Global Agnostic	DiffFacto (Ours)
SNAP ( $\downarrow$ )	25.24	18.23	19.29	<b>13.32</b>

Table 4: **Ablation on our Factorization.** Snapping metric on three connections for chairs: back to leg or seat; seat to legs; and arms to seat or back. ( $CD \times 10^{-2}$ ).

Direct reg.	cVAE [45]	cGAN [36]	cIMLE [30] (Ours)
13.38	7.33	11.48	<b>4.97</b>

Table 5: **Multi-modality of Transformation Sampler.** Shape inversion on the chair category ( $CD \times 10^{-4}$ ).

## 5.5. Ablations

**Factorized Joint Prior.** We ablate our joint factorized prior  $P(\mathbf{Z}, \mathbf{T})$  by replacing it with separate independent part distributions (*Separate*). Each part distribution  $P(S_j)$  is modeled with a separate CNF prior. Tab. 4 shows that our approach achieves better inter-part score.

**Cross Diffusion.** We also evaluate several variants of our cross diffusion network, termed as follows: *Post Transform* – we remove the cross diffusion network and instead model the conditional likelihood of points only on part styles, then separately applying the part transformations as a post-processing step. *Global Agnostic* – we remove the cross attention component that provides global shape information through the  $m$  tokens, and model only the point distribution  $P(X|z_j, \tau_j, j)$  for each part. Tab. 4 shows that our approach also achieves better inter-part level score.

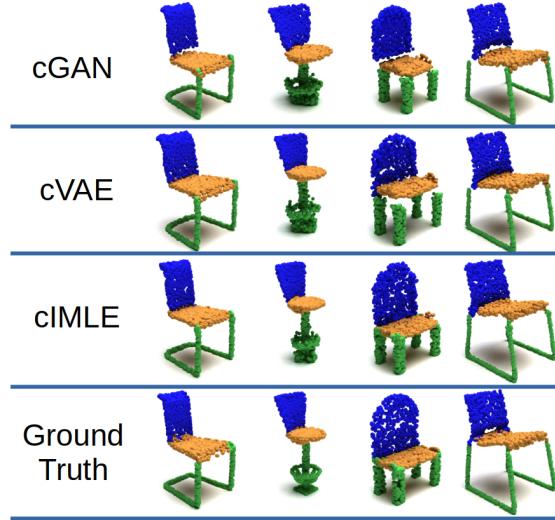
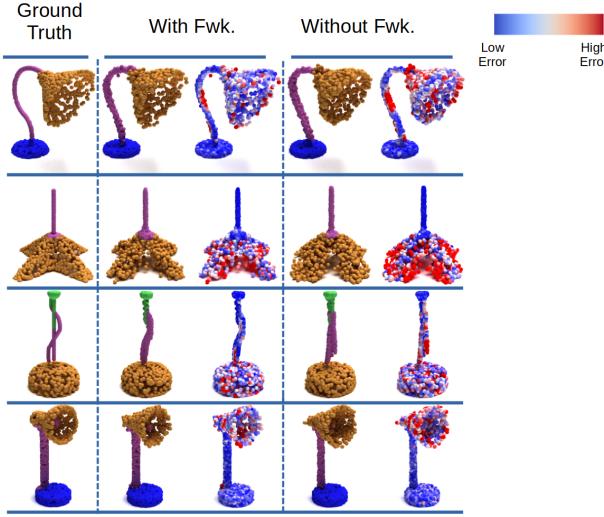


Figure 7: **Shape Inversion Comparison.** We show shape inversion examples using different implicit probabilistic methods to model the transformation distribution given part style latents.

**Transformation Sampler.** We ablate our transformation

sampler that uses cIMLE [30] and compare it with direct regression of  $\tau$ , as well as unimodal cVAE [45] and cGAN [36] on shape inversion. Tab. 5 quantitatively shows that our approach achieves the best results with a large margin and Fig. 7 shows examples of inverted shapes using different implicit methods. Notice that across all examples, cIMLE is able to recover most accurately the correct transformation through inversion because it models a multimodal distribution where different modes can be recovered during the sampling process.

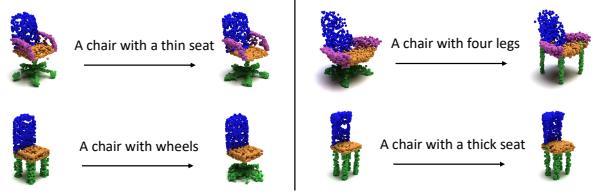


**Figure 8: Qualitative Comparison w. Generalized Forward Kernel.** Examples of reconstructed lamps with our generalized kernel (**With Fwk.**) versus without the generalized kernel (**Without Fwk.**). For each example, we show the reconstructed shape (**Ground Truth**) where each point is colored by the minimum distance to points in the ground truth shape.

**Generalized Forward Kernel.** Our generalized forward kernel works by modeling the diffusion prior as a transformed Gaussian distribution that captures informative positional and scale information of the part. Fig. 6 shows that such a kernel allows better transformation extrapolation, where geometry is better preserved on extreme user edits. Moreover, Fig. 8 shows qualitative examples of the advantages of using our generalized forward kernel as illustrated by the heat map on the per-point reconstruction error. We note that our generalized forward kernel is able to model complex part geometry better than the standard forward kernel (see the cap on rows 1, 2, and 4 in the figure) because of the additional size and location prior information incorporated into the diffusion process.

## 6. Future Works and Limitations

Our method by design requires segmented shapes for training, as our cross diffusion network and generalized forward kernel require a hard assignment of points to the



**Figure 9: Potential Language Guided Edits.** We show prototypes of potential future work where we can edit existing shapes via our part style latent distributions with language guidance. The left sides are input shapes and we update the part latent vectors based on the language inputs. The resulting shapes are shown on the right side of the arrows.

semantic part of the shape, *i.e.* it conditions on label  $j$ . Although this allows us to learn smooth latent spaces for each part, as well as global shape distribution  $P(\mathbf{S})$ , it constrains us in training with datasets that are semantically labeled. A future direction would be a formulation that enables soft assignments, which would allow us to train on unsegmented data.

Another promising future direction is to enable language-driven editing through our proposed factorization. Our factorization allows for localized edits, where language can be used to identify which semantic part,  $j$ , to edit as well as the direction to modify its corresponding latent code,  $Z_j$ , to adhere with the language description. As a preliminary demonstration, we use the dataset from ShapeGlot [2] that consists of a triplet of shapes with a corresponding language description. We then select a negative shape and edit it using DiffFacto in order to match with the corresponding text description. A match is enforced by optimizing the edit to match the positive example in the triplet. Fig. 9 shows some visual examples.

## 7. Conclusion

In this paper we propose DiffFacto, a deep generative probabilistic framework for generating 3D point clouds in a controllable manner. By factorizing out the shape distribution into individual semantically-meaningful parts plus their transformations, we allow for intuitive control over the generated shapes. The framework is also flexible and readily available for various applications ranging from style mixing to configuration editing.

**Acknowledgements.** This work is supported by ARL grant W911NF-21-2-0104, a Vannevar Bush Faculty Fellowship, the Natural Science Foundation of China (Project No. 62220106003), Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology, and the Natural Sciences and Engineering Research Council of Canada. We very much appreciate Congyue Deng for her helpful discussion at the early stage of this project. We are also grateful for the advice and help from Colton Stearns and Davis Rempe.

# Supplementary Materials

This document supplements our submission DiffFacto: Controllable Part-Based 3D Point Cloud Generation with Cross Diffusion. In particular, we provide detailed derivations and proofs of our training objective (Sec S1) and generalized forward kernel (Sec S2), additional experiment comparisons and results (Sec S3), and additional details on our network (Sec S4), implementation (Sec S5) and experiment set-ups (Sec S6).

## Contents

<b>1. Introduction</b>	<b>1</b>
<b>2. Related Work</b>	<b>2</b>
<b>3. Problem Overview</b>	<b>3</b>
<b>4. Method</b>	<b>3</b>
4.1. Training Objective . . . . .	4
4.2. Factorized Joint Prior $P(\mathbf{Z}, \mathbf{T})$ . . . . .	4
4.3. Cross Diffusion Network . . . . .	5
4.4. Enabling Shape Editing . . . . .	5
<b>5. Results</b>	<b>6</b>
5.1. Dataset and Evaluation Protocol . . . . .	6
5.2. Baselines . . . . .	7
5.3. Controllable Shape Generation . . . . .	7
5.4. Shape Editing . . . . .	7
5.5. Ablations . . . . .	8
<b>6. Future Works and Limitations</b>	<b>9</b>
<b>7. Conclusion</b>	<b>9</b>
<b>S1Objective Derivation</b>	<b>11</b>
<b>S2Generalized Forward Diffusion Kernel</b>	<b>12</b>
<b>S3Additional Experimental Results</b>	<b>13</b>
S3.1Additional Quantitative Comparison . . . . .	13
S3.2Autoencoding . . . . .	14
S3.3Ablation on Part Stylizer . . . . .	14
S3.4Additional Part Style Mixing Results . . . . .	14
S3.5Additional Part Level Interpolation Results . . . . .	14
S3.6Additional Part Level Sampling Results . . . . .	15
S3.7Additional Part-Configuration Sampling Results . . . . .	15
<b>S4Network Details</b>	<b>39</b>
S4.1Part Style Sampler . . . . .	39
S4.2Transformation Sampler . . . . .	39
S4.3Cross Diffusion Network . . . . .	40
<b>S5Implementation Details</b>	<b>40</b>
S5.1Training Details . . . . .	40
S5.2Network Parameters . . . . .	41
S5.2.1 Part Style Sampler . . . . .	41
S5.2.2 Transformation Sampler . . . . .	41
S5.2.3 Cross Diffusion Network . . . . .	41

S5.3Algorithm . . . . .	42
<b>S6Experiment Set-up Additional Details</b>	<b>42</b>
S6.1Control-Enabled Baselines Additional Details . . . . .	42
S6.2Evaluation Protocol Additional Details . . . . .	42
S6.2.1 Intra-part Evaluation Additional Details . . . . .	42
S6.2.2 Inter-Part Evaluation Additional Details . . . . .	44
S6.2.3 Human Study Additional Details . . . . .	44

## S1. Objective Derivation

We derive the evidence lower bound presented in Sec. 4.1 of our main paper. Let  $Q_\varphi(\mathbf{Z}, \mathbf{T}|S)$  be a variational family with the following factorization

$$Q_\varphi(\mathbf{Z}, \mathbf{T}|S) = Q_\varphi(\mathbf{Z}|S) Q(\mathbf{T}|S) = Q(\mathbf{T}|S) \prod_{j=1}^m Q_{\varphi_j}(Z_j|S_j) = Q(\mathbf{T}|S) \prod_{j=1}^m Q_{\varphi_j}(Z_j|\hat{S}_j), \quad (10)$$

where we learn independent  $Q_{\varphi_j}(Z_j|\hat{S}_j) \forall j$ , i.e. our part stylizers, as a variational encoder that is parameterized as a Gaussian distribution with learnable mean and diagonal variance. Since given an input segmented shape  $S$ , its corresponding part transformations  $\mathbf{T}$  is known, then the distribution  $Q(\mathbf{T}|S)$  is deterministic. Furthermore, we highlight that each of the encoders  $Q_{\varphi_j}$  only takes in the canonicalized part  $\hat{S}_j$  as input, which enables us to model the prior distribution  $P(Z_j)$  to only encode information about the part style and not of its size and scale. With this variational family, we can write the evidence lower bound as:

$$\begin{aligned} & \mathbb{E}_S [\log P_{\phi, \psi, \theta}(S)] \\ &= \mathbb{E}_S \left[ \log \iint P_\phi(S|\mathbf{z}, \boldsymbol{\tau}) P_{\psi, \theta}(\mathbf{z}, \boldsymbol{\tau}) d\mathbf{z} d\boldsymbol{\tau} \right] \\ &= \mathbb{E}_S \left[ \log \iint \frac{P_\phi(S|\mathbf{z}, \boldsymbol{\tau}) P_\psi(\mathbf{z}) P_\theta(\boldsymbol{\tau}|\mathbf{z})}{Q_\varphi(\mathbf{z}, \boldsymbol{\tau}|S)} Q_\varphi(\mathbf{z}, \boldsymbol{\tau}|S) d\mathbf{z} d\boldsymbol{\tau} \right] \\ &= \mathbb{E}_S \left[ \log \iint \frac{P_\phi(S|\mathbf{z}, \boldsymbol{\tau}) P_\theta(\boldsymbol{\tau}|\mathbf{z}) \prod_{j=1}^m P_{\psi_j}(z_j)}{Q(\boldsymbol{\tau}|S) \prod_{j=1}^m Q_{\varphi_j}(z_j|\hat{S}_j)} Q(\boldsymbol{\tau}|S) \prod_{j=1}^m Q_{\varphi_j}(z_j|\hat{S}_j) d\mathbf{z} d\boldsymbol{\tau} \right] \\ &= \mathbb{E}_S \left[ \log \mathbb{E}_{z_j \sim Q_{\varphi_j}, \boldsymbol{\tau} \sim Q(\boldsymbol{\tau}|S)} \left[ \frac{P_\phi(S|\mathbf{z}, \boldsymbol{\tau}) P_\theta(\boldsymbol{\tau}|\mathbf{z}) \prod_{j=1}^m P_{\psi_j}(z_j)}{Q(\boldsymbol{\tau}|S) \prod_{j=1}^m Q_{\varphi_j}(z_j|\hat{S}_j)} \right] \right] \\ &\quad \text{By Jensen's Inequality,} \\ &\geq \mathbb{E}_{S \sim P(S), z_j \sim Q_{\varphi_j}, \boldsymbol{\tau} \sim Q(\boldsymbol{\tau}|S)} \left[ \log \frac{P_\phi(S|\mathbf{z}, \boldsymbol{\tau}) P_\theta(\boldsymbol{\tau}|\mathbf{z}) \prod_{j=1}^m P_{\psi_j}(z_j)}{Q(\boldsymbol{\tau}|S) \prod_{j=1}^m Q_{\varphi_j}(z_j|\hat{S}_j)} \right] \\ &= \mathbb{E}_{S, z_j, \boldsymbol{\tau}} \left[ \log P_\phi(S|\mathbf{z}, \boldsymbol{\tau}) + \sum_{j=1}^m \log \frac{P_{\psi_j}(z_j)}{Q_{\varphi_j}(z_j|\hat{S}_j)} + \log \frac{P_\theta(\boldsymbol{\tau}|\mathbf{z})}{Q(\boldsymbol{\tau}|S)} \right] \end{aligned}$$

Since  $Q(\mathbf{T}|S)$  is a deterministic distribution,  $Q(\boldsymbol{\tau}|S) = 1$  for all  $\boldsymbol{\tau} \sim Q(\boldsymbol{\tau}|S)$ . Thus,

$$\begin{aligned} &= \mathbb{E}_{S, z_j, \boldsymbol{\tau}} \left[ \log P_\phi(S|\mathbf{z}, \boldsymbol{\tau}) + \sum_{j=1}^m \log \frac{P_{\psi_j}(z_j)}{Q_{\varphi_j}(z_j|\hat{S}_j)} + \log P_\theta(\boldsymbol{\tau}|\mathbf{z}) \right] \\ &= \mathbb{E}_{S, \mathbf{z}, \boldsymbol{\tau}} [\log P_\phi(S|\mathbf{z}, \boldsymbol{\tau})] - \sum_{j=1}^m \mathbb{E}_S [\text{KL}(P_{\psi_j}(z_j) \| Q_{\varphi_j}(z_j|\hat{S}_j))] + \mathbb{E}_{S, \mathbf{z}, \boldsymbol{\tau}} [\log P_\theta(\boldsymbol{\tau}|\mathbf{z})]. \end{aligned} \quad (11)$$

As detailed in the main paper, each term in the final objective corresponds to the training loss for each of our components (part stylizers, transformation sampler, and cross diffusion network). The first term is the reconstruction error given prior information  $\mathbf{z}, \tau$ , and its maximization is done by the cross diffusion network, which is a Denoising Diffusion Probabilistic Model with a generalized forward diffusion kernel. We elaborate on the derivation of the generalized forward kernel in Sec. S2, and the cross diffusion network in Sec. S4.3. The second term is the KL divergence regularization loss for each of the part style latent distribution  $P_{\psi_j}(Z_j)$ . We implement each of the part style latent distributions with a prior flow model. For specifics, please refer to Sec. S4.1. Lastly, the third/final term corresponds to the part transformations, where the cIMLE training strategy maximizes this the objective, as explained in Sec. S4.2.

## S2. Generalized Forward Diffusion Kernel

We derive the forward diffusion process with the proposed generalized forward kernel in our Cross Diffusion Network (Sec. 4.3 of the main paper). Given a probability distribution  $Q(X^{(0)})$  with  $X^{(0)} \in \mathbb{R}^d$  that we want to model, we can define a forward diffusion process with a mean  $\mu \in \mathbb{R}^d$  and a variance  $\Sigma \in \mathbb{R}_{>0}^{d \times d}$  such that

$$Q(X^{(0:T)}|\mu, \Sigma) = Q(X^{(0)}) \prod_{t=1}^T Q(X^{(t)}|X^{(t-1)}, \mu, \Sigma)$$

where

$$Q(X^{(t)}|X^{(t-1)}, \mu, \Sigma) = \mathcal{N}\left(\sqrt{\alpha_t}X^{(t-1)} + (1 - \sqrt{\alpha_t})\mu, (1 - \alpha_t)\Sigma\right)$$

for each  $t = 1, \dots, T$ .  $\alpha_t = 1 - \beta_t$  are variance scheduling parameters. Notice that by setting  $\Sigma = \mathbf{I}$  and  $\mu = \mathbf{0}$  we recover the classical diffusion kernel. Similar to the classical kernel, we can efficiently sample from  $Q(X^{(t)}|x^{(0)}, \mu, \Sigma)$  using the reparameterization trick:

$$Q\left(X^{(t)}|X^{(0)}, \mu, \Sigma\right) = \mathcal{N}\left(\sqrt{\bar{\alpha}_t}X^{(0)} + (1 - \sqrt{\bar{\alpha}_t})\mu, (1 - \bar{\alpha}_t)\Sigma\right).$$

Here,  $\bar{\alpha}_t = \prod_{t'=1}^t \alpha_{t'}$ . As  $t \rightarrow \infty$  increases,  $\bar{\alpha}_t$  decreases to zero. Thus, for large  $T$ , the distribution  $Q(X^{(t)}|X^{(0)}, \mu, \Sigma)$  approaches the Gaussian distribution  $\mathcal{N}(\mu, \Sigma)$  with mean  $\mu$  and variance  $\Sigma$ . By doing so, we can incorporate prior information in the forward diffusion process in terms of  $\mu$  and  $\Sigma$ . We can also show that for each  $t > 1$ , the posterior distribution  $Q(X^{(t-1)}|X^{(t)}, X^{(0)}, \mu, \Sigma)$  is also a Gaussian distribution, given by

$$Q\left(X^{(t-1)}|X^{(t)}, X^{(0)}, \mu, \Sigma\right) = \mathcal{N}\left(\Xi_t, \eta_t^2 \Sigma\right), \quad (12)$$

where

$$\begin{aligned} \Xi_t &= \frac{\beta_t \sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_t} X^{(0)} + \frac{(1 - \bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1 - \bar{\alpha}_t} X^{(t)} + \left(1 + \frac{(\sqrt{\alpha_t} - 1)(\sqrt{\alpha_t} + \sqrt{\bar{\alpha}_{t-1}})}{1 - \bar{\alpha}_t}\right) \mu, \\ \eta_t^2 &= \frac{\beta_t(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}. \end{aligned}$$

We can further simplify the expression for  $\Xi_t$  by substituting

$$X^{(t)} = \sqrt{\bar{\alpha}_t}X^{(0)} + (1 - \sqrt{\bar{\alpha}_t})\mu + (1 - \bar{\alpha}_t)\varepsilon\sqrt{\Sigma}, \quad (13)$$

where  $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  and  $\sqrt{\Sigma}$  is the Cholesky decomposition of  $\Sigma$ . Thus, similar to the classical diffusion parameterizations, we can define a reverse diffusion process  $P_\phi(X^{(0:T)}|\mu, \Sigma)$  that is also a Markov chain so that

	Chair			Airplane		
	MMD-P ( $\downarrow$ )	COV-P ( $\uparrow$ )	1NNA-P	MMD-P ( $\downarrow$ )	COV-P ( $\uparrow$ )	1NNA-P
SP-GAN [31]	4.43	31.7	87.77	5.27	26.2	91.03
SPAG [18]	4.53	36.3	78.94	5.73	26.3	90.17
NW [21]	4.77	34.0	83.36	4.77	34.0	83.36
<b>DiffFacto (Ours)</b>	<b>3.27</b>	<b>42.5</b>	<b>65.23</b>	<b>3.20</b>	<b>46.2</b>	<b>68.72</b>

Table S1: **Intra-part evaluation with additional baselines.** MMD-P score is multiplied by  $10^{-2}$ . COV-P and 1NNA-P are reported in %

$$P_\phi(X^{(0:T)}|\mu, \Sigma) = P(X^{(T)}|\mu, \Sigma) \prod_{t=1}^T P_\phi(X^{(t-1)}|x^{(t)}, \mu, \Sigma),$$

where each  $P_\phi(X^{(t-1)}|x^{(t)}, \mu, \Sigma)$  is a Gaussian distribution parameterized by weights  $\phi$  and tries to approximate the ground truth posterior distribution  $Q(X^{(t-1)}|X^{(t)}, X^{(0)}, \mu, \Sigma)$ . Specifically, as in classical diffusion models [20], we can parameterize each reverse kernel as a Gaussian distribution with learnable mean and fixed variance:

$$P_\phi(X^{(t-1)}|x^{(t)}, \mu, \Sigma) = \mathcal{N}\left(\Xi_\phi\left(t, x^{(t)}, \mu, \Sigma\right), \eta_t \sqrt{\Sigma}\right)$$

for each  $t > 1$ . In practice, instead of directly learning  $\Xi_\phi$ , we can parameterize  $\Xi_\phi$  using Eq (12) and learn a noise approximator  $\varepsilon_\phi$  that approximate random noises. At sampling time, we can recover  $P_\phi(X^{(t-1)}|x^{(t)}, \mu, \Sigma)$  using  $\varepsilon_\phi$  by writing

$$\begin{aligned} \Xi_\phi\left(t, x^{(t)}, \mu, \Sigma\right) &= \frac{\beta_t \sqrt{\alpha_{t-1}}}{1 - \bar{\alpha}_t} X^{(0)} + \frac{(1 - \bar{\alpha}_{t-1}) \sqrt{\alpha_t}}{1 - \bar{\alpha}_t} X^{(t)} + \left(1 + \frac{(\sqrt{\alpha_t} - 1)(\sqrt{\alpha_t} + \sqrt{\bar{\alpha}_{t-1}})}{1 - \bar{\alpha}_t}\right) \mu \\ &= \frac{1}{\sqrt{\alpha_t}} x^{(0)} - \frac{1 - \sqrt{\alpha_t}}{\sqrt{\alpha_t}} \mu - \frac{\beta_t \varepsilon_\phi(t, x^{(t)}, \mu, \Sigma)}{\sqrt{\alpha_t}(1 - \bar{\alpha}_t)} \sqrt{\Sigma}. \end{aligned} \quad (14)$$

Doing so, the training objective of the generalized diffusion process simplifies to an L2 loss with noise, as we discussed in our main paper.

### S3. Additional Experimental Results

We provide additional experiment results in this section. We show additional quantitative comparisons in Sec. S3.1. In Sec. S3.2, we evaluate DiffFacto’s autoencoding quality compared with state-of-the-art point cloud reconstruction networks, as well as demonstrate the advantage of the generalized forward kernel in our Cross Diffusion Network in modeling complex part geometry and topology. In Sec. S3.3, we ablate the prior flows in our part style sampler. In Sec. S3.4 and Sec. S3.5, we showcase shape mixing and part-level interpolation results compared with control-enabled baselines. Lastly, in Sec. S3.6 and Sec. S3.7, we showcase additional qualitative results on part-level and transformation sampling.

#### S3.1. Additional Quantitative Comparison

We provide additional quantitative comparisons for DiffFacto on part-level generation as shown in Tab. S1. Specifically, we include two state-of-the-art mesh-based generative networks SPAGHETTI [18] and Neural Wavelets [21]. As shown, although these methods require 3D mesh supervision, our approach is able to achieve better part-level generative scores compared to these works. Additionally, we also compare with SP-GAN [31], which is a GAN-based point cloud generative network. We also note that unlike the other point cloud baselines, SP-GAN does not allow for encoding of an existing shape, making shape editing non-trivial if not infeasible. We used pre-trained weights for the three networks and follow the same evaluation procedure as for the other baselines.

Dataset	Metrics	DPM	DPM <sup>†</sup>	ShapeGF	ShapeGF <sup>†</sup>	Ours*	Ours (DiffFacto)
Airplane	CD ( $\downarrow$ )	2.266	1.70	2.082	1.599	1.460	<b>1.413</b>
	EMD ( $\downarrow$ )	1.96	1.72	2.258	1.722	1.337	<b>1.333</b>
Car	CD ( $\downarrow$ )	7.686	7.013	7.016	6.735	6.559	<b>6.538</b>
	EMD ( $\downarrow$ )	3.229	3.179	3.283	3.323	<b>2.782</b>	2.788
Chair	CD ( $\downarrow$ )	5.297	4.39	4.521	3.928	3.746	<b>3.701</b>
	EMD ( $\downarrow$ )	2.83	2.64	3.181	2.646	2.222	<b>2.214</b>
Lamp	CD ( $\downarrow$ )	8.334	5.55	8.243	5.100	3.787	<b>3.481</b>
	EMD ( $\downarrow$ )	3.395	2.899	4.504	2.983	1.959	<b>1.951</b>

Table S2: Comparison of point cloud auto-encoding performance.  $\dagger$  represents a modification of baselines to incorporate part latents. \* represents our framework without modification of the forward kernel. Reported CD and EMD are multiplied by  $10^4$  and  $10^2$  respectively.

### S3.2. Autoencoding

We report autoencoding performance of DiffFacto in Tab. S2 compared to DPM [34] and ShapeGF [5]. We further modify the two baselines to model part latents by also encoding each segmented part. We also report their reconstruction score denoted by  $\dagger$  superscript. We see that DiffFacto outperforms all baselines by a margin due to our novel cross diffusion network and generalized forward kernel. We also note that there is an increase in performance for both baselines with the additional part latent modification  $\dagger$ . We also report the reconstruction score for our model without the generalized forward kernel, marked as *Ours\**. Notice that there is a larger gap between our approach with and without the generalized forward kernel in the lamp category in which parts exhibit more complex geometrical and topological structures.

### S3.3. Ablation on Part Stylizer

	MMD-P	COV-P	1NNA
VAE	3.47	<b>44.3</b>	72.59
CNF (Ours)	<b>3.27</b>	42.5	<b>65.23</b>

Table S3: **Ablation on Part Stylizer.** Individual-part level metrics ablating our part stylizer ( $\text{MMD-P} \times 10^{-2}$ ).

We replace the CNF in our part stylizer by directly using a standard Gaussian as the prior, *i.e.* VAE. Tab. S3 shows intra-part evaluation metrics comparing CNF with standard VAE. With the lower 1NNA and MMD-P, we note that CNF gives an overall better performance in capturing the part-style distribution than with vanilla VAE. We also observe that shapes generated with CNF are overall much clearer and with less noise than with VAE. This is because the prior flow relaxes the latent space regularization constraint by allowing the prior distribution to be transformed into a more complex distribution. For details of the prior flow, see Sec. S4.1.

### S3.4. Additional Part Style Mixing Results

Since we sample from independent part-style distributions to generate coherent shapes, shape mixing is a direct application where modeling the factorized latent distribution has a direct advantage. We showcase additional part style mixing results in chairs, airplanes, and lamps categories in Fig. S2, Fig. S1, and Fig. S3. We also further evaluate our mixing quality compared to our control-enabled baselines, Ctrl-LION [59] and Ctrl-ShapeGF [5], for the chair and airplane categories. Because we model a distribution of parts’ size and location given a set of part style latent codes, across all the examples, we are able to produce coherent shapes despite the input parts having different sizes and locations in their original source shapes. In particular, when compared to the control-enabled baselines that do not explicitly model parts sizes, the baselines’ mixing results often produce shapes with detached or interpenetrating parts as a result of incorrect part configuration.

### S3.5. Additional Part Level Interpolation Results

Moreover, since we explicitly model a part-style latent space, we can also interpolate shapes on the part level. Given two samples of part style codes  $z_j, z'_j \sim P_{\psi_j}(Z_j)$ , we use linear interpolation from  $z_j$  to  $z'_j$  obtain intermediate part latents

uniformly spaced in between:  $z_j^k = z_j + \frac{k}{10}(z'_j - z_j)$ . We showcase qualitative results in all four categories in Fig. S4, S5, S6, S7 and S10. Notice that in all four categories, our interpolation results show smooth transitions from the source to the target of the interpolation, as most highlighted in the chair leg interpolation in Fig. S7. Moreover, the intermediate shapes are also plausible, where discrete jumps can be observed when no intermediate configuration is available. An example of this can be seen in the interpolation of engines in Fig. S4, where the number of engines jumps from four to two to allow a discrete transition. Moreover, we see that the unreferenced parts (colored in grey) remain unchanged during the interpolation process because we model independent distributions of part styles. Overall, the part-level interpolation results showcase our models' ability to generate plausible shapes with fine-grained control.

We also qualitatively compare DiffFacto with the control-enabled baselines, Ctrl-LION [59] and Ctrl-ShapeGF [5], on part-level interpolation quality. Fig. S11 and S13 show part level interpolation result for Ctrl-LION and Fig. S12 and S14 show part level interpolation result for Ctrl-ShapeGF, for the chair and airplane categories. We follow the same part-level interpolation procedure on the control-enabled baselines as described earlier for our method. Since the control enabled baselines do not explicitly model the distribution of part transformations given their styles, the interpolation of each part may result in detached parts. See the interpolation on chairs in Fig. S13. Moreover, since Ctrl-ShapeGF generates shapes by training a cGAN directly on its autoencoding latent space, its interpolation is less smooth as the autoencoding latent space is trained on discrete training examples. This can also be seen from the chair interpolation in Fig. S14 (see the last row where the intermediate geometry exhibits extensive artifacts).

### S3.6. Additional Part Level Sampling Results

We showcase that DiffFacto enables part-based controllable generation through part-level sampling. We include additional results on part-style sampling to supplement the left and middle column of Figure 3 in the main paper. We showcase both *sampling* a part style and *fixing* a part style for each shape. Fig. S15, S17 and S19 show additional qualitative results for *sampling* individual part styles (Figure 3 main paper -left). Fig. S16, S18 and S20 show additional qualitative results for *fixing* individual part styles (Figure 3 main paper -middle).

### S3.7. Additional Part-Configuration Sampling Results

We further show additional qualitative examples on our ability to enable part-configuration sampling, *i.e.* generate various plausible configurations of the shape with fixed part styles. Fig. S21 to S24 show additional qualitative results. This is supplement the right column in Figure 3 of the main paper.

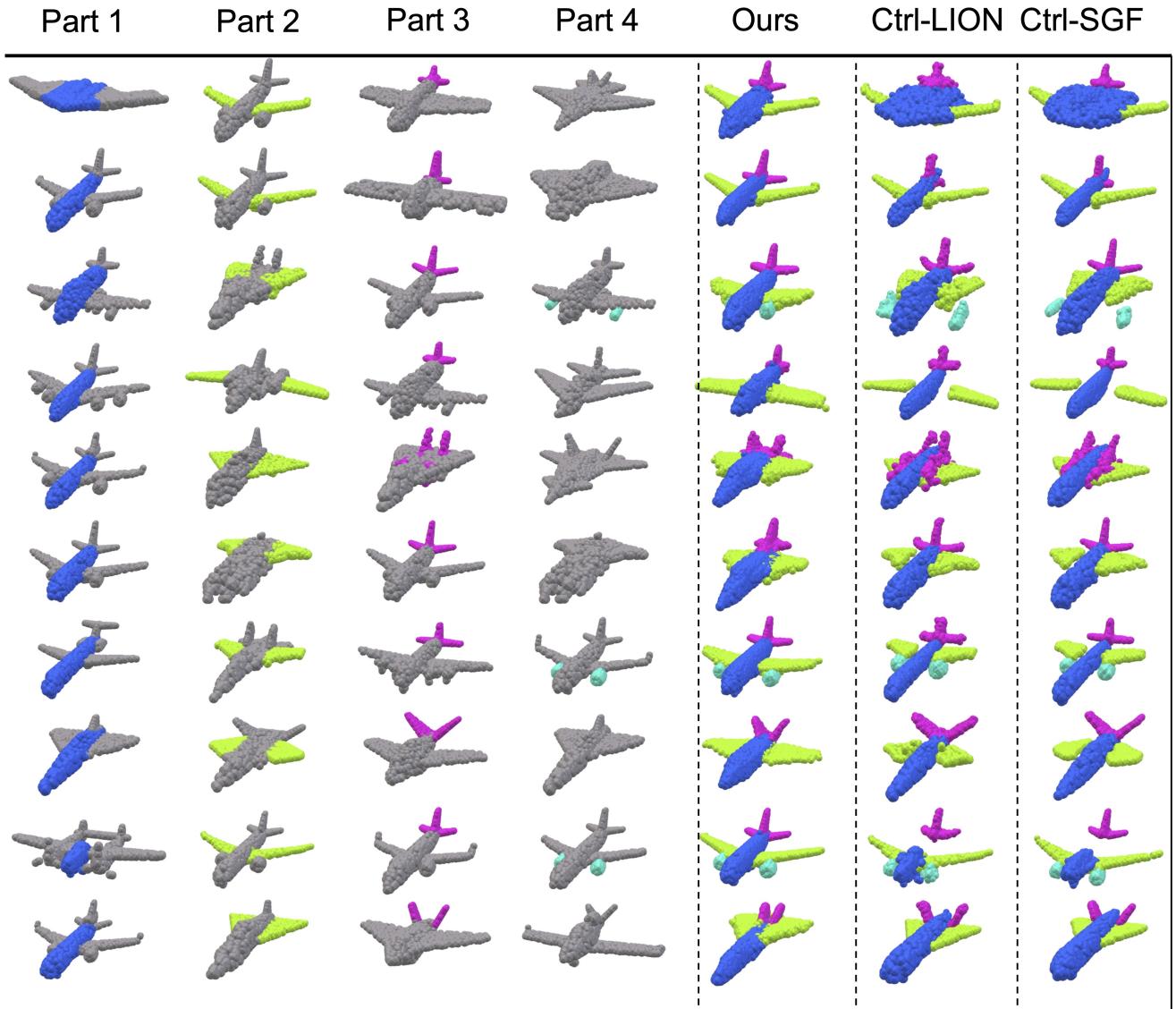


Figure S1: **Shape Mixing: Airplanes.** For each row, parts from different shapes are encoded and mixed for both our method (**Ours**) and the control-enabled baselines (**Ctrl-LION**, **Ctrl-SGF**). Notice that the control-enabled baselines fail to produce coherent shapes because it does not model the distribution of valid transformations while our method transforms each part globally into a coherent configuration so that the generated shapes are plausible.

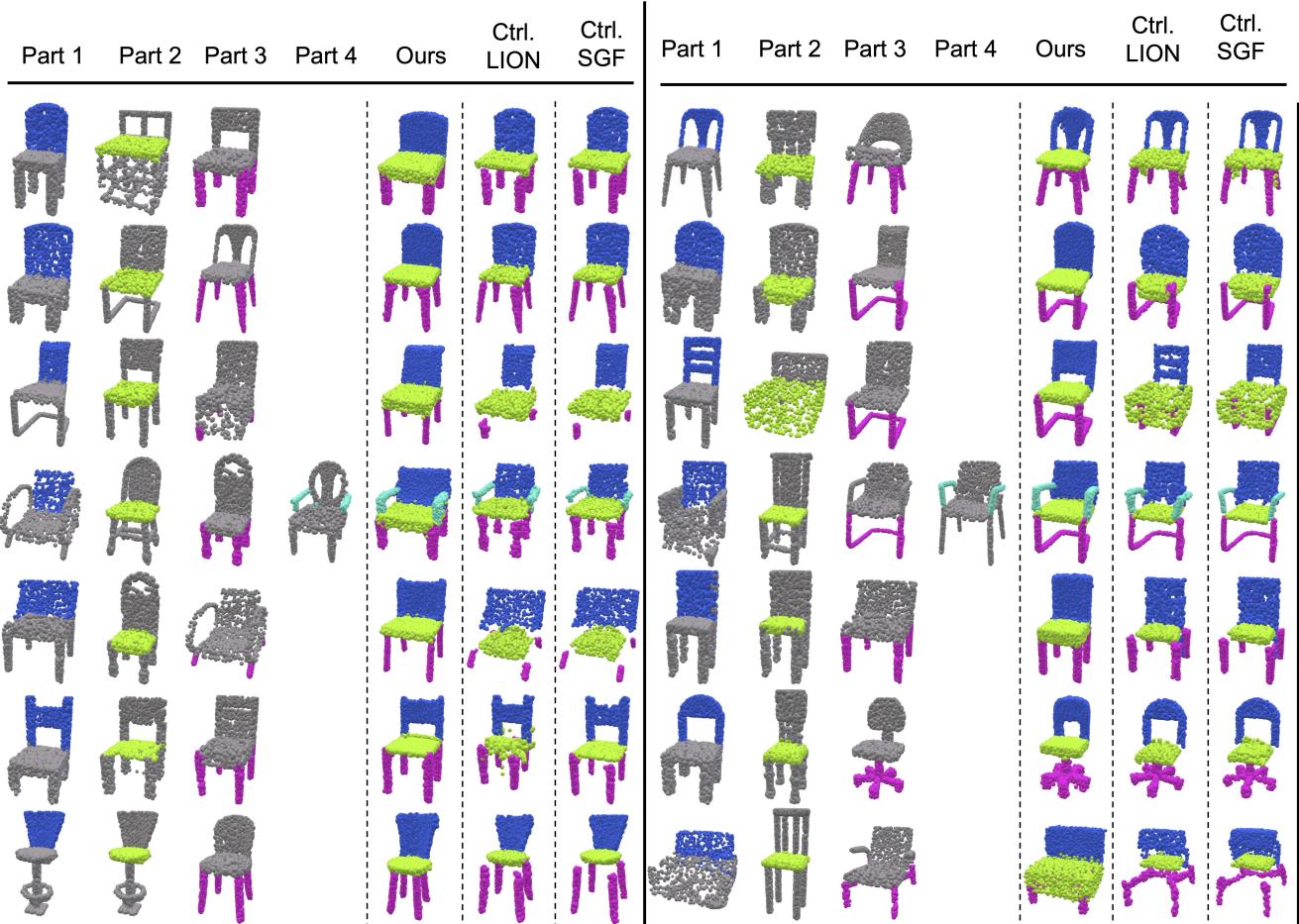


Figure S2: **Shape Mixing: Chairs.** For each example (two examples per row, separated by the solid line), parts from different shapes are encoded and mixed for both our method (**Ours**) and the control-enabled baselines (**Ctrl-LION**, **Ctrl-SGF**). Notice that the control-enabled baselines fail to produce coherent shapes because it does not model the distribution of valid transformations while our method transforms each part globally into a coherent configuration so that the generated shapes are plausible.

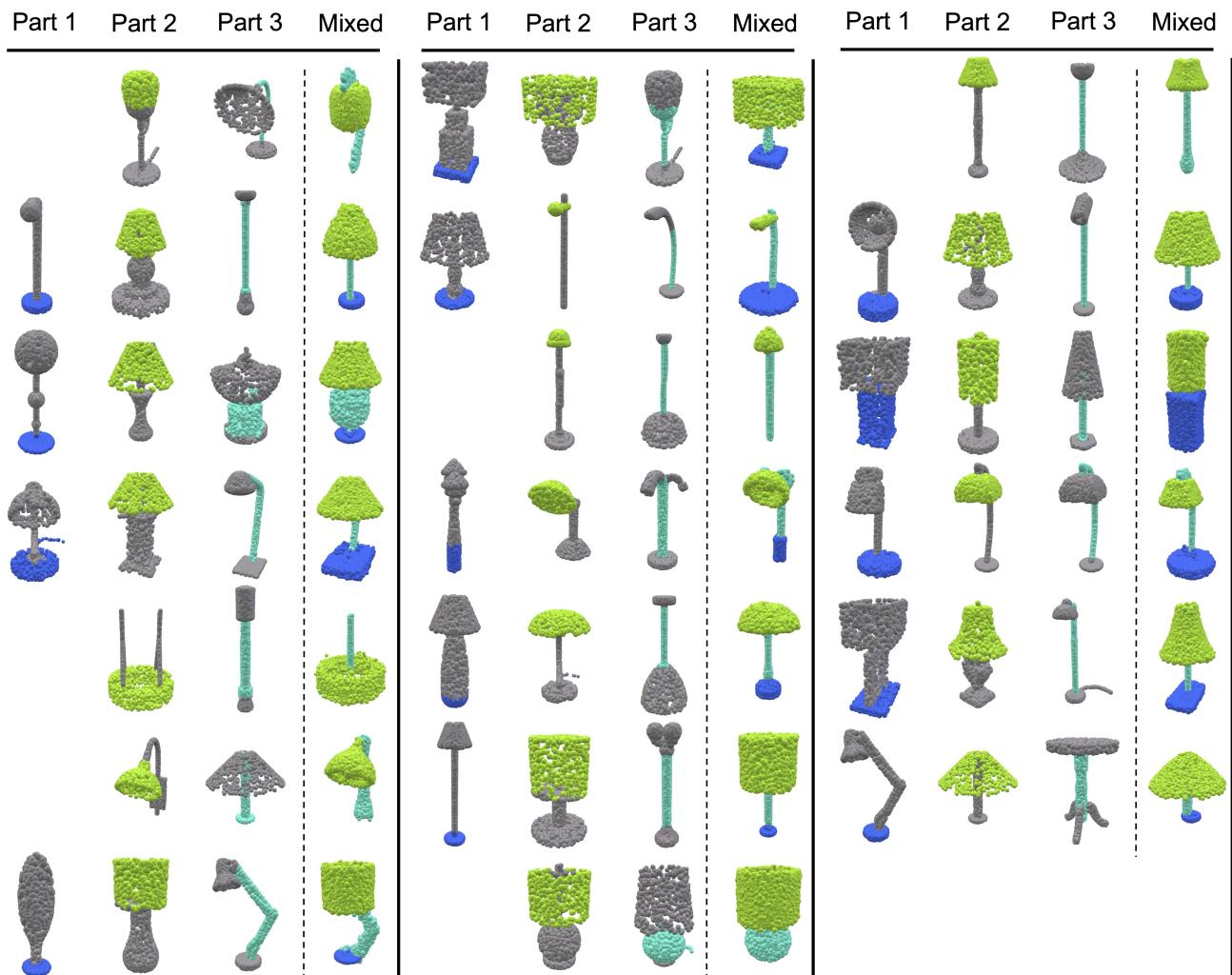


Figure S3: **Shape Mixing: Lamps.** Each row shows three examples of shape mixing for lamps. Notice that our model can adjust the parts' size and location so that the mixed shape is still coherent.

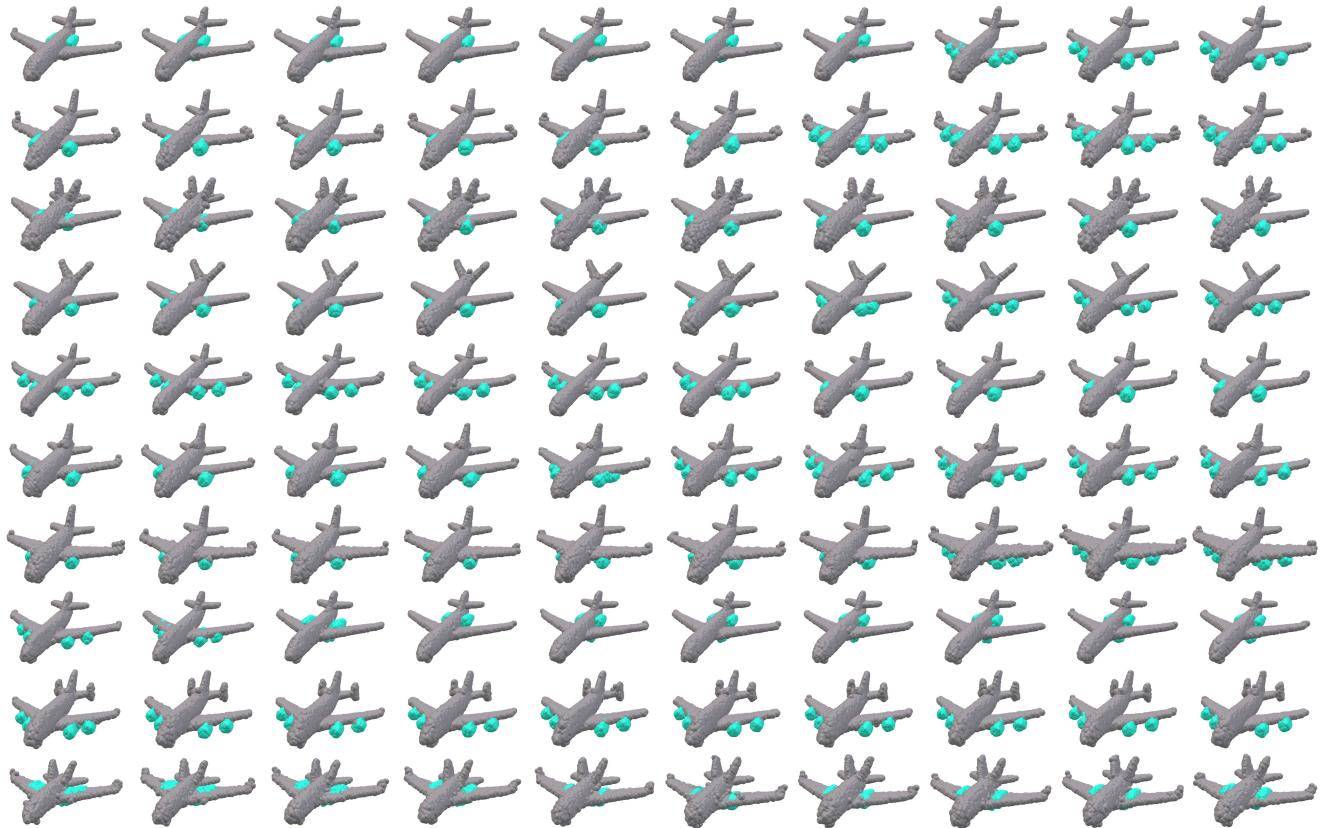


Figure S4: **Airplane Part Level Interpolation: Engines.** Each row is one interpolation path of the engine (colored part). Notice that because of the discrete nature of engine's number and location, the interpolation path between a different number of engines, and different locations of engines shows distinctive jumps from one configuration to another. This implies that the part latent space learned by our model only generates plausible shapes.

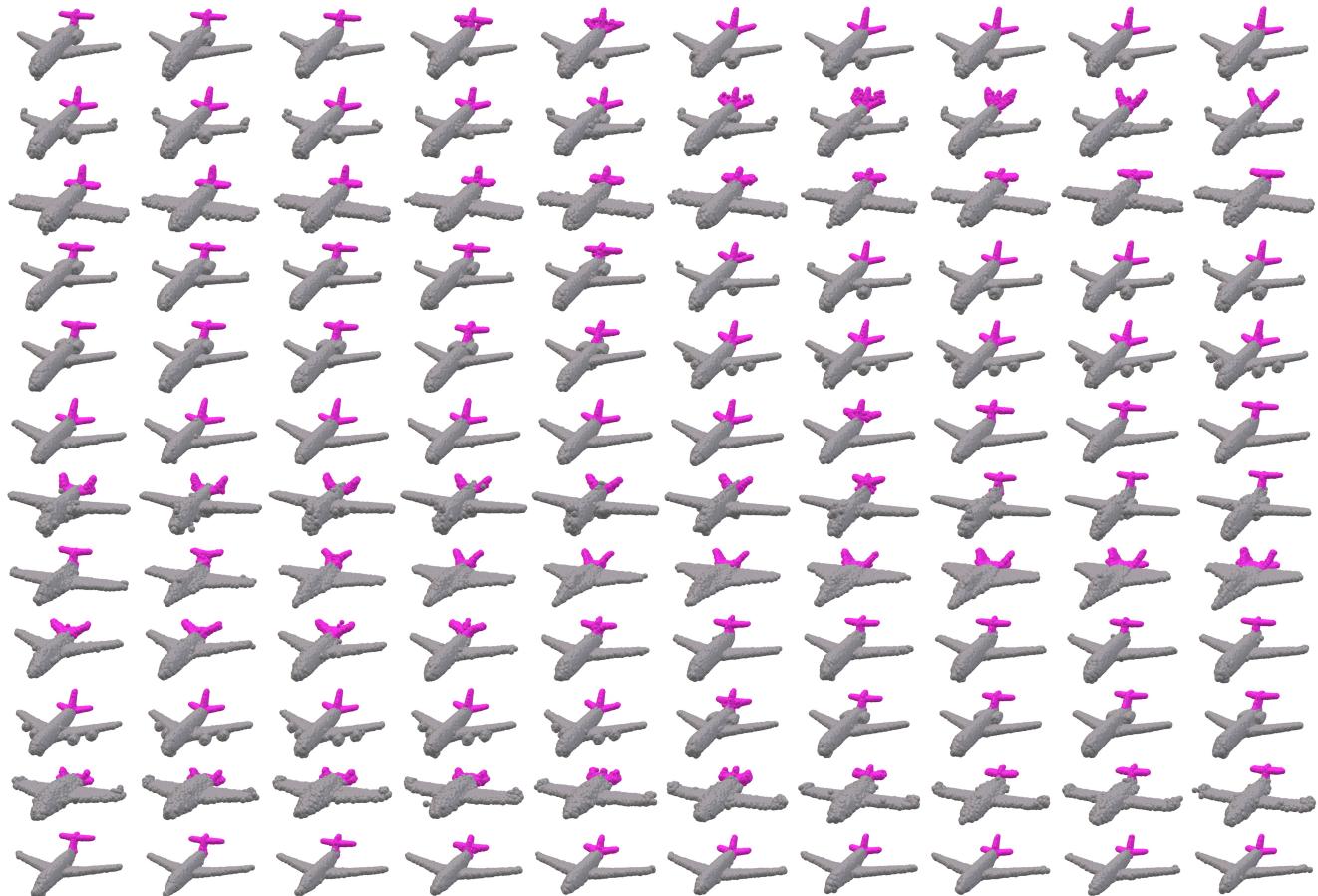


Figure S5: **Airplane Part Level Interpolation: Tail Wings.** Each row is one interpolation path of the tail wing (colored part). Similar to the engine's interpolation, the tail exhibits distinctive jumps from one mode to another so that the intermediate shapes are all plausible and coherent.

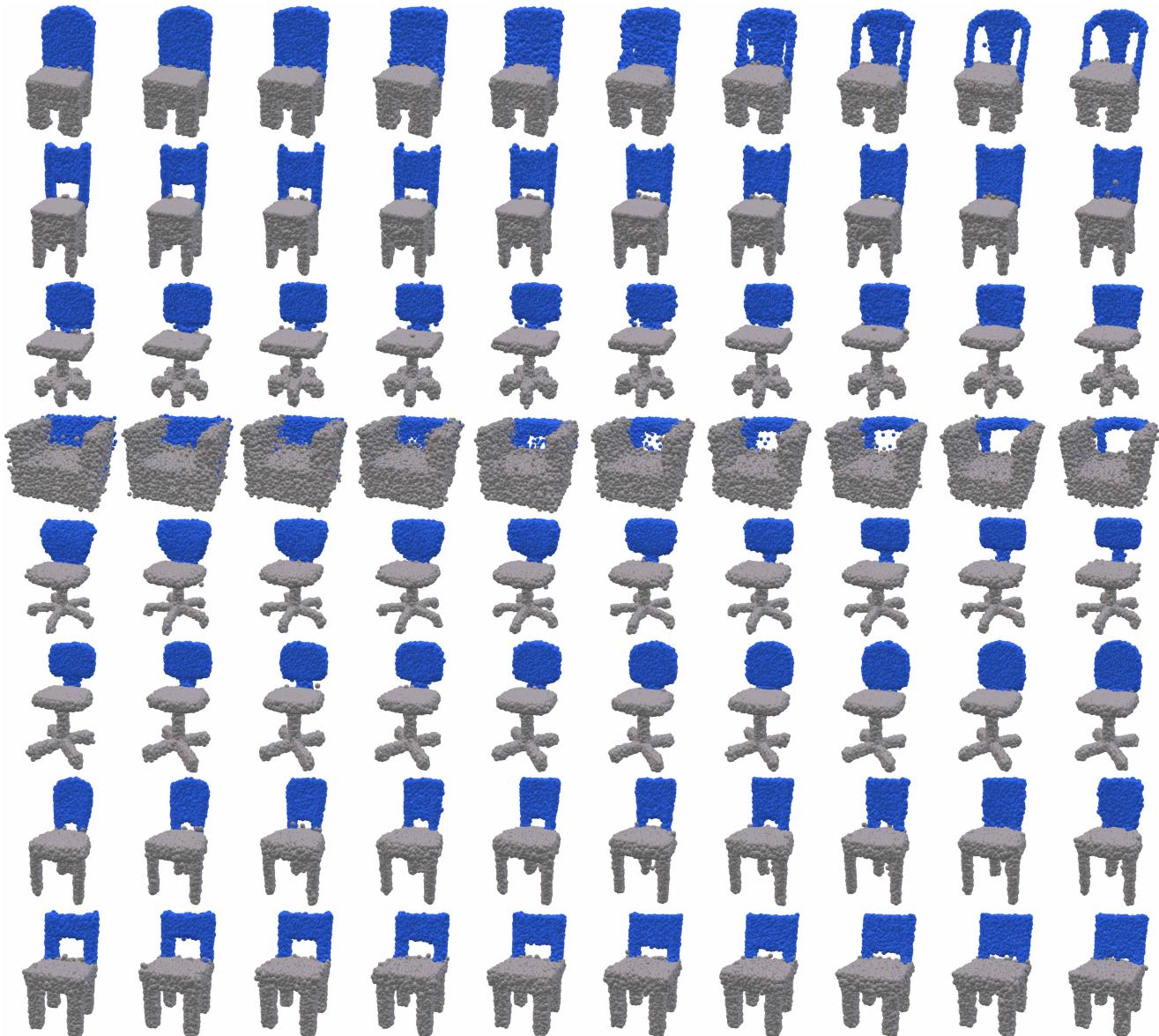


Figure S6: **Chair Part Level Interpolation: Back.** Each row is one interpolation path of the back (colored part). Notice that to adjust for the change in the back's style during interpolation, the unreferenced parts (in grey) adjust their size and location so that the output shapes are still coherent (see rows 3 and 6 for such covariance).



Figure S7: **Chair Part Level Interpolation: Legs.** Each row is one interpolation path of the legs (colored part). Notice that the intermediate shapes exhibit little artifacts and the interpolation path smoothly transitions between two different styles of legs. See, for example, rows 2 and 6.

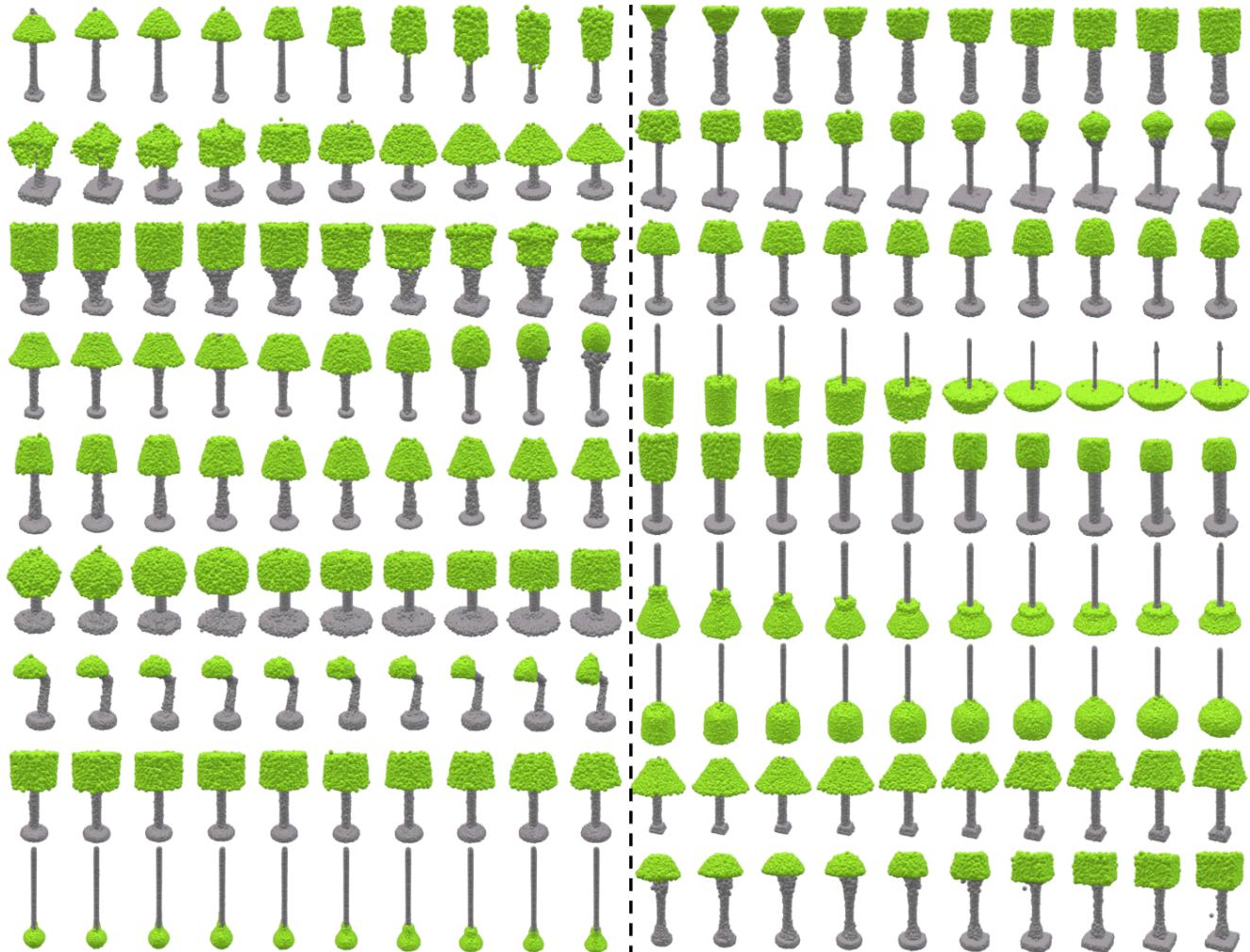


Figure S8: **Lamp Part Level Interpolation: Lamp Cap.** Each row shows two interpolation paths of the lamp cap (colored part) separated by the dashed line. Notice that for some of the interpolations where the starting part geometry is different from the ending geometry, the pole (in grey) needs to adjust its size so that the generated shape is coherent.

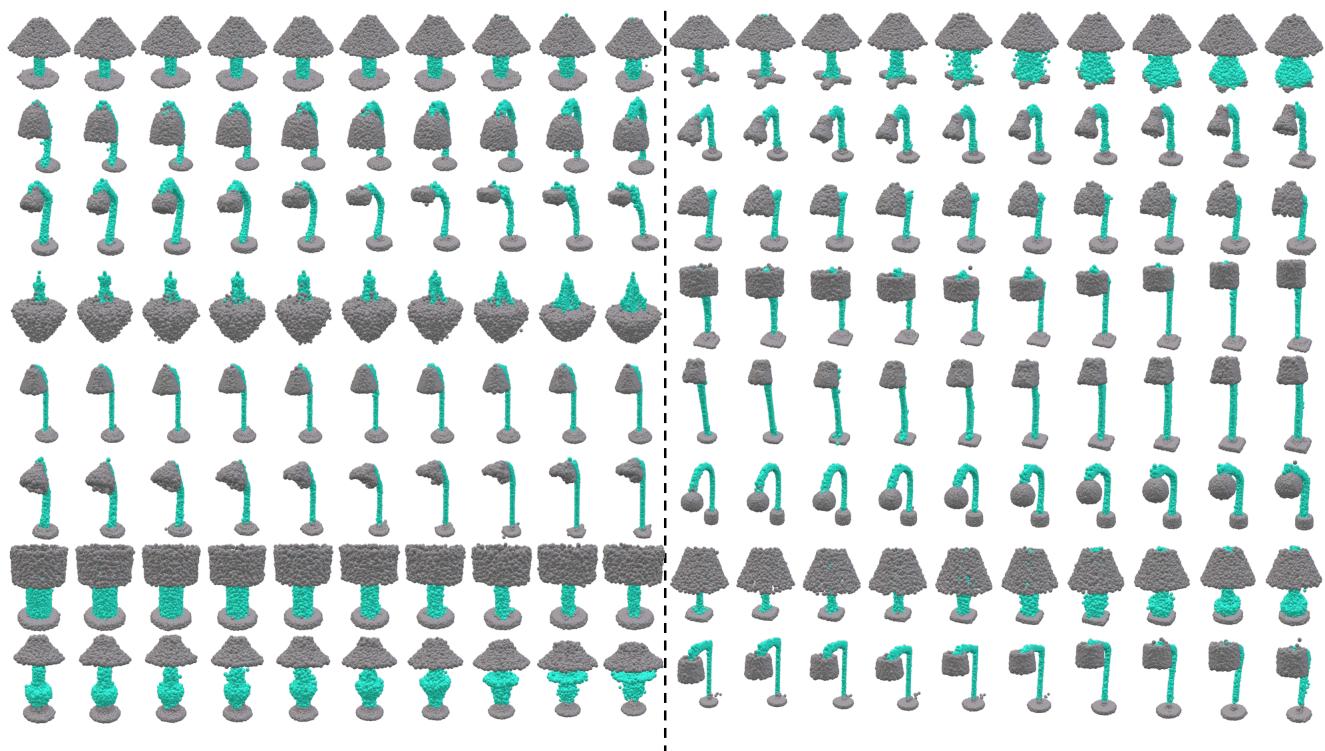


Figure S9: **Lamp Part Level Interpolation: Lamp Pole.** Each row shows two interpolation paths of the lamp pole (colored part) separated by the dashed line. Notice that sometimes the size of the lamp cap needs to vary accordingly to fit the change in the pole's style. See, for example, interpolations on rows 2 and 4.

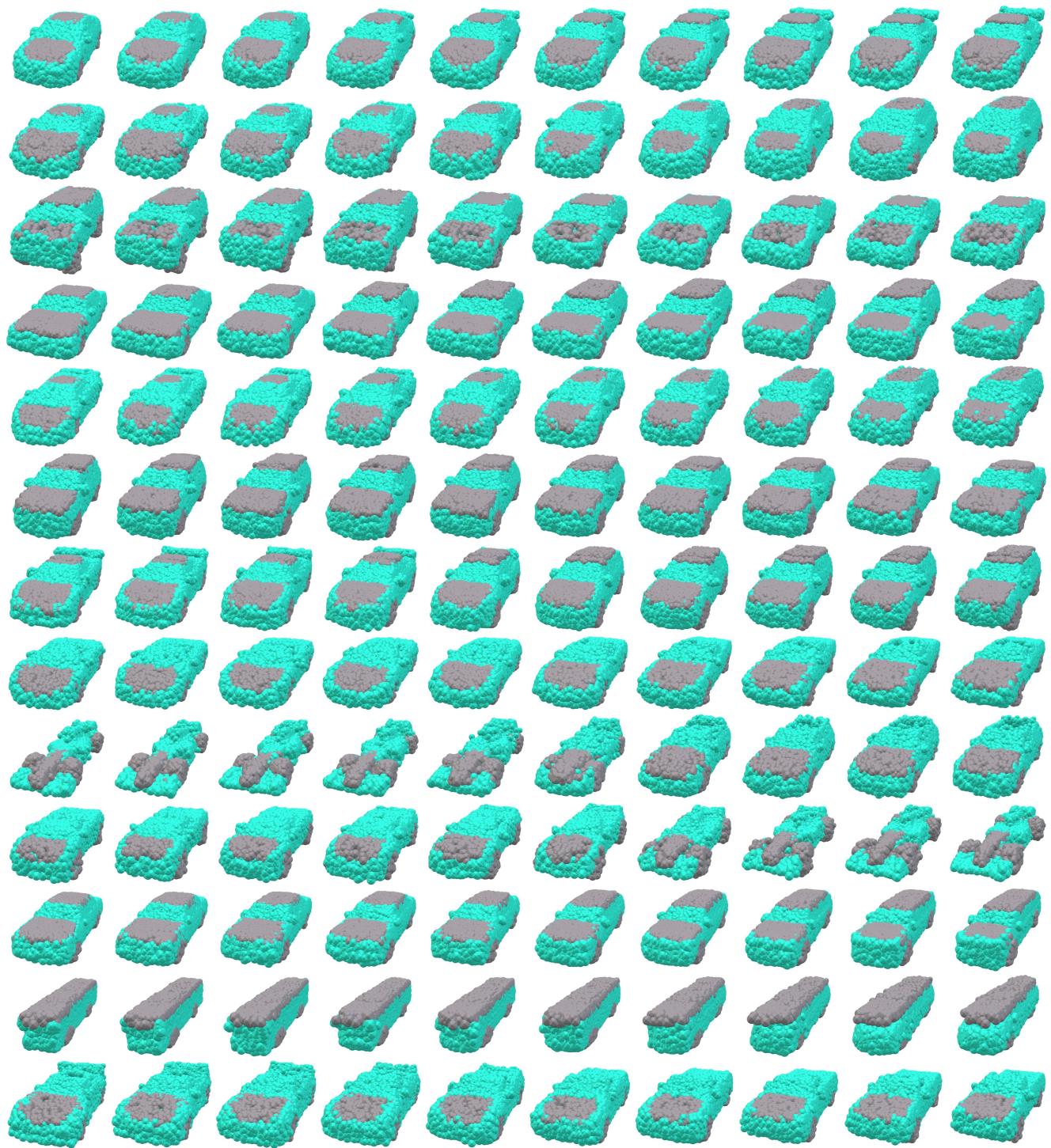


Figure S10: **Car Part Level Interpolation: Body** Each row shows one interpolation path of the car's body (colored part). Notice that the unreferenced parts vary accordingly to adjust to the change in the body's style (See rows 9 and 12).

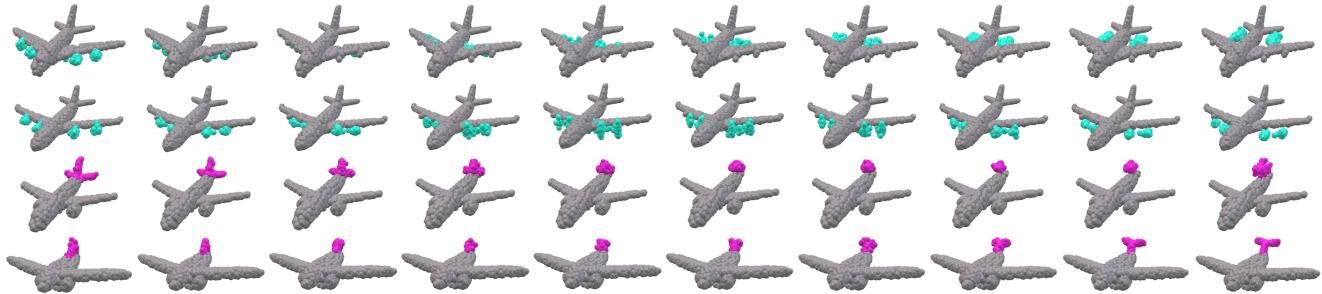


Figure S11: **Control LION Part Level Interpolation: Airplanes.** Each row shows one interpolation path of the colored part. Notice that since the control enabled LION does not model the distribution of part configurations, the intermediate shapes during interpolation often become invalid and not coherent. See, for example, row 3.

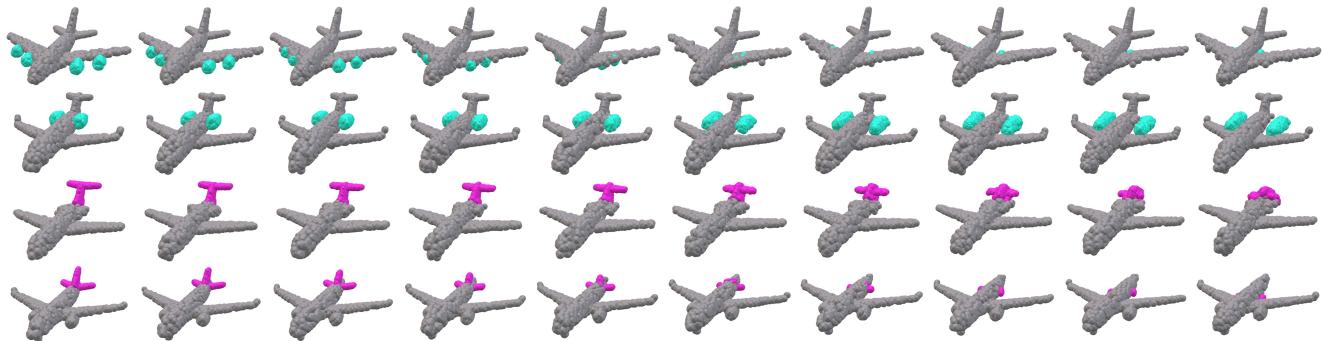


Figure S12: **Control ShapeGF Part Level Interpolation: Airplanes.** Each row shows one interpolation path of the colored part. Since the control enabled ShapeGF samples part latents on its autoencoding latent space, the interpolation path often shows non-smoothness in its intermediate steps. See, for example, the last row.

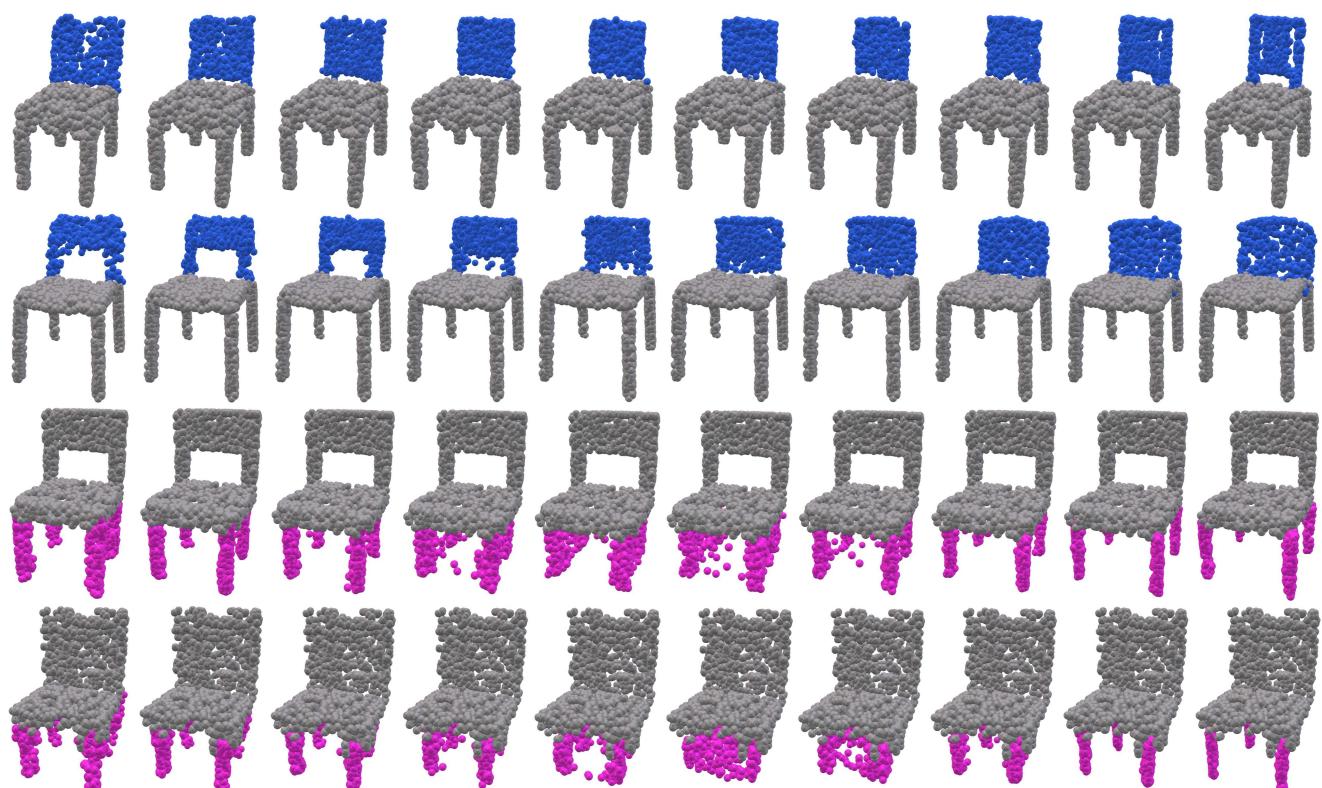


Figure S13: **Control LION Part Level Interpolation: Chairs.** Each row shows one interpolation path of the colored part. Since the control-enabled LION does not model the distribution of part configuration, the intermediate shapes often become detached (see the first row for such an example).

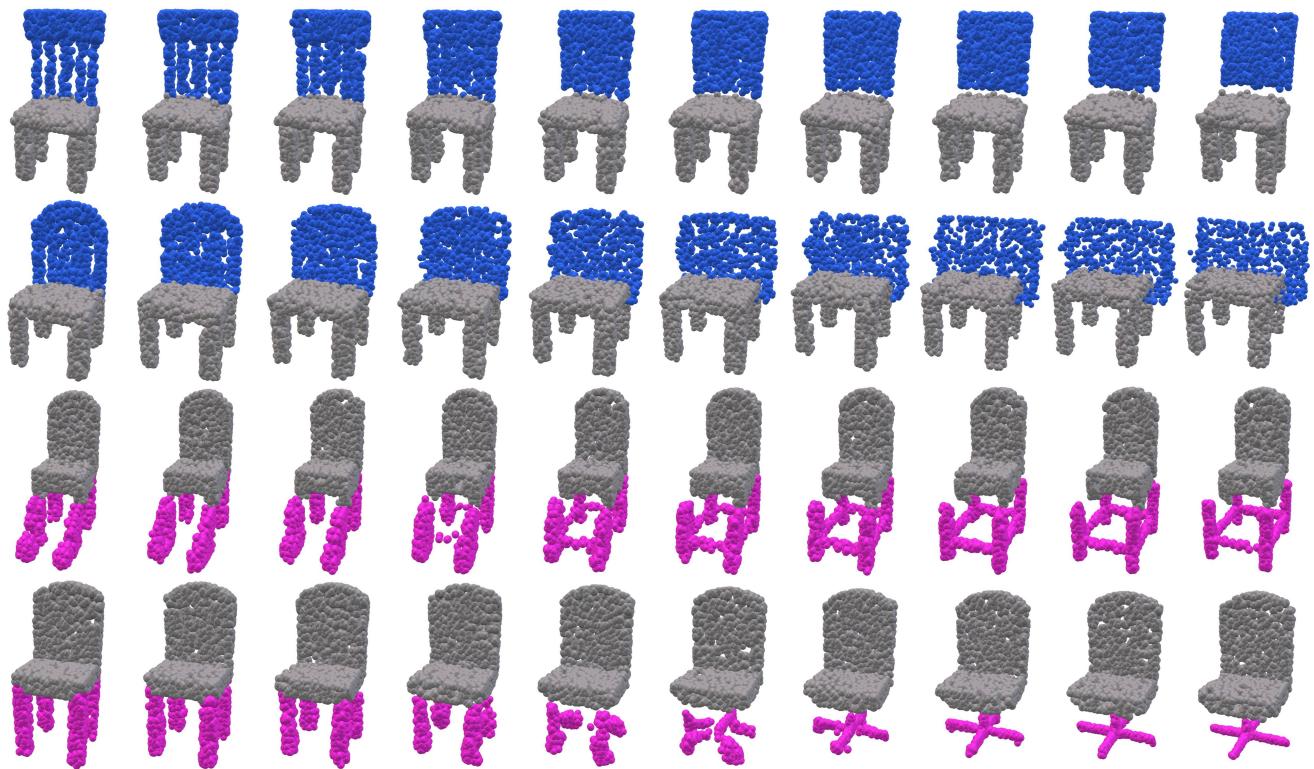
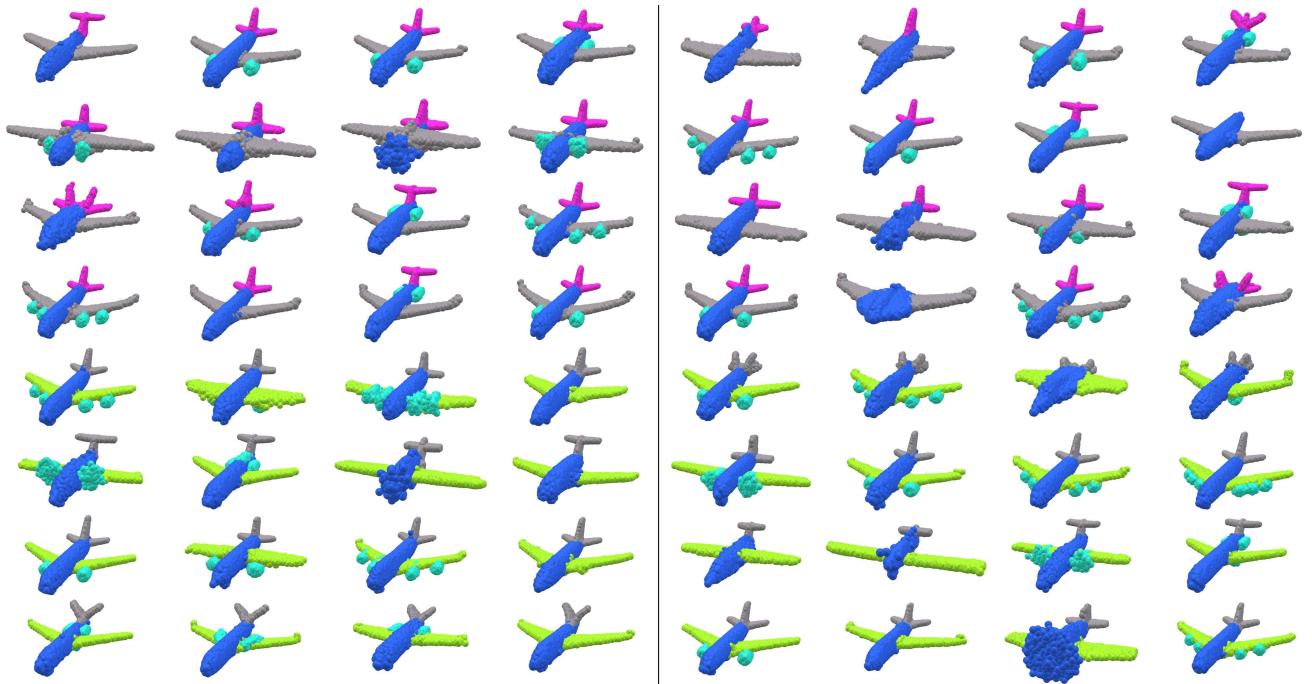


Figure S14: **Control ShapeGF Part Level Interpolation: Chairs.** Each row shows one interpolation path of the colored part. The last row shows the control-enabled ShapeGF exhibits artifacts in its interpolated shapes. Furthermore, parts become detached because it does not model the distribution of part transformation.



Figure S15: **Part-level Sampling: Airplanes, Sampling one part.** Each row shows two sets of airplanes (separated by the solid line) generated by sampling the part style latent associated with the colored part while keeping the rest of the part style latents for the grey parts fixed. Notice that the grey parts' styles stay fixed while the colored part varies across different samples. This ability to sample on the part level allows for intuitive user control in shapes generated.



**Figure S16: Part-level Sampling: Airplanes, Fixing one part.** Each row shows two sets of airplanes (separated by the solid line) generated by sampling the set of part style latents associated with the colored parts while keeping the part style latent for the grey part fixed. Notice that the grey part's style stays fixed while the colored parts vary across different samples. The size and location of the grey part adjust to the sampled other part styles to generate coherent shapes.

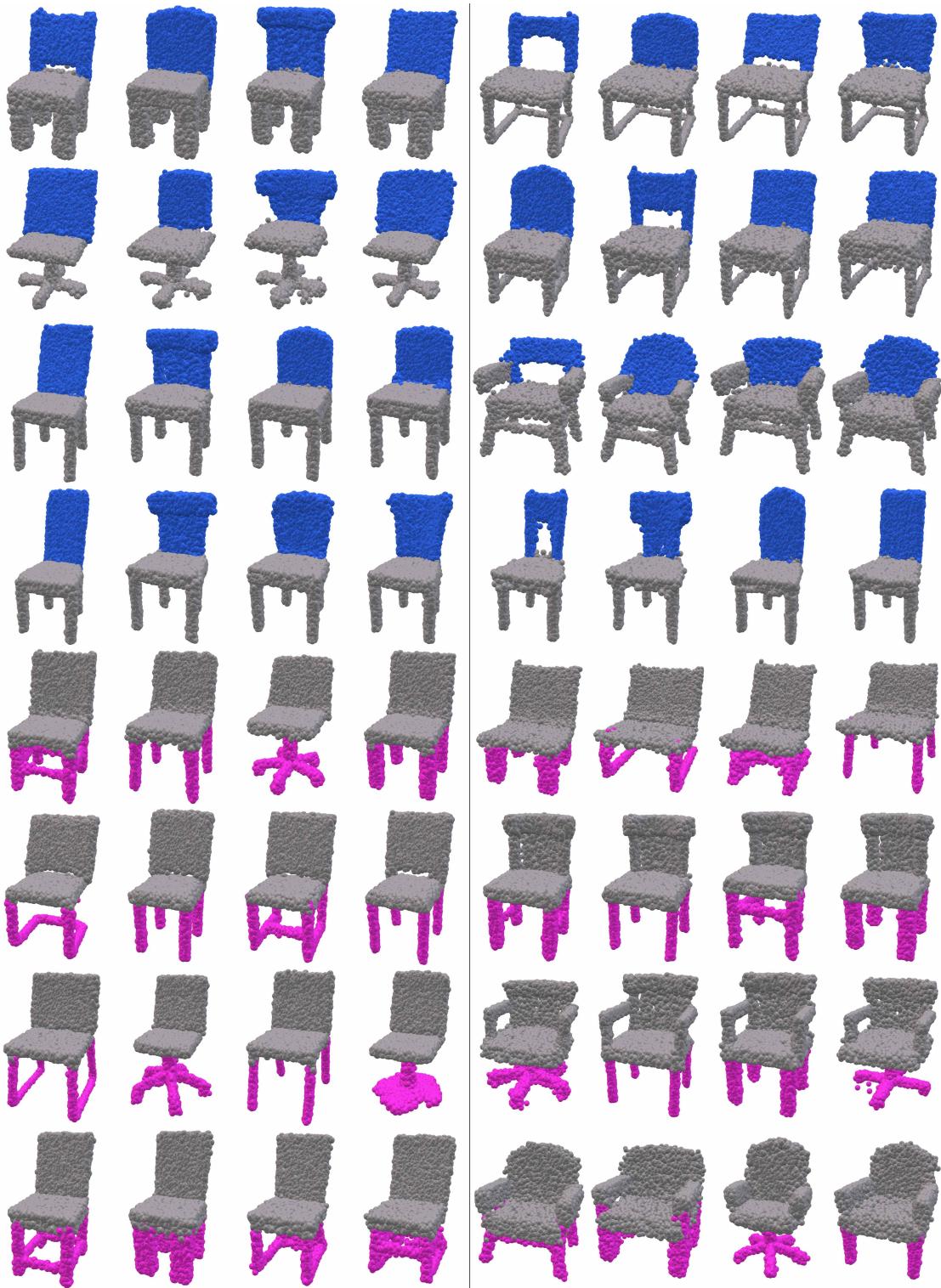


Figure S17: **Part-level Sampling: Chairs, Sampling one part.** Each row shows two examples (separated by the solid line) of sampling the colored part style latent while keeping the other part style latents unchanged. Notice that, based on the sampled part style, the unreferenced part changes their sizes accordingly so that the output shape stays plausible and coherent. See, for example, the change of seat's size on rows 2 and 3.

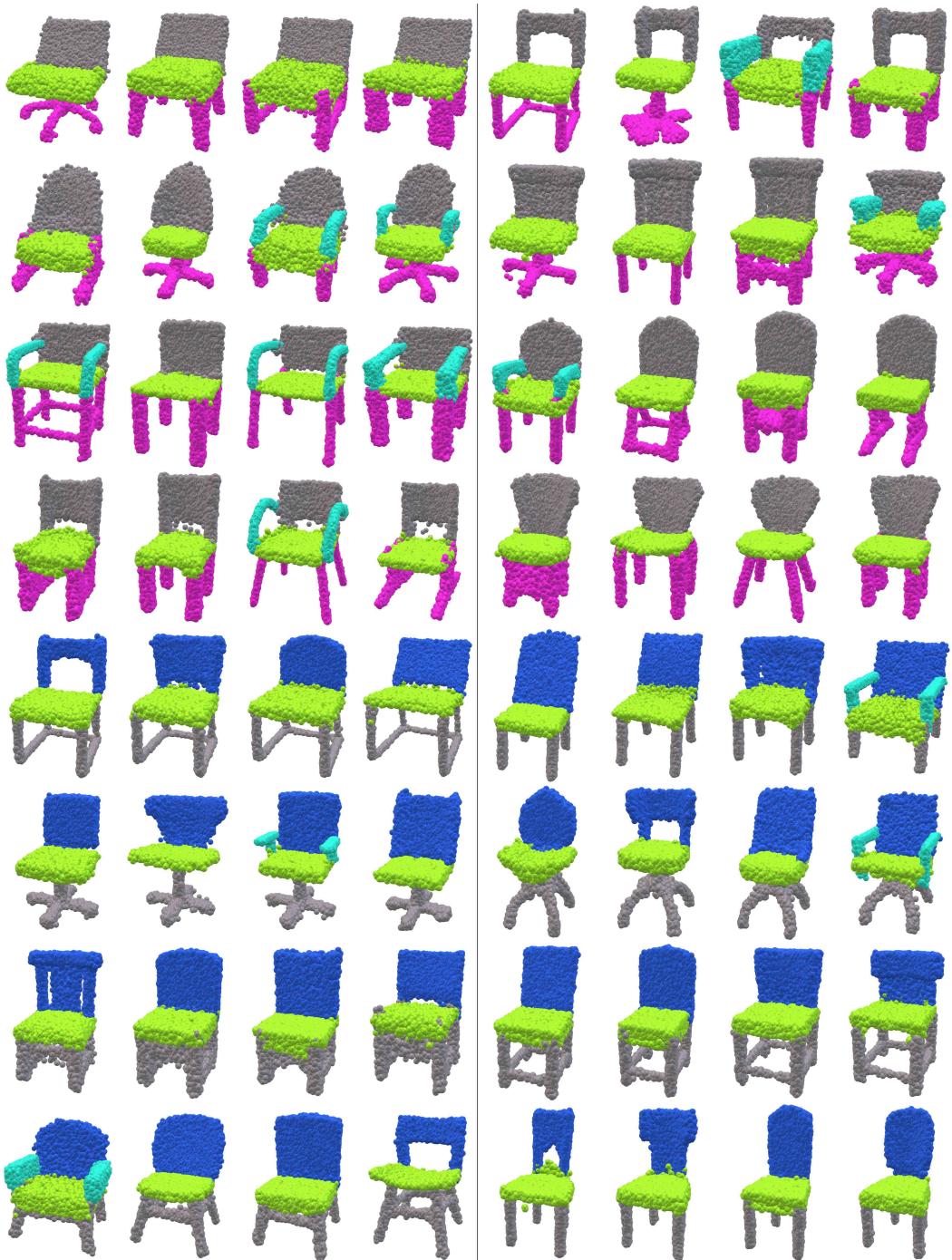


Figure S18: **Part-level Sampling: Chairs, Fixing one part.** Each row shows two sets of chairs (separated by the solid line) generated by sampling the set of part style latents associated with the colored parts while keeping the part style latent for the grey part fixed. Because we model the part configurations independently from their styles, the generated shapes are all plausible and coherent while showing desired controls.

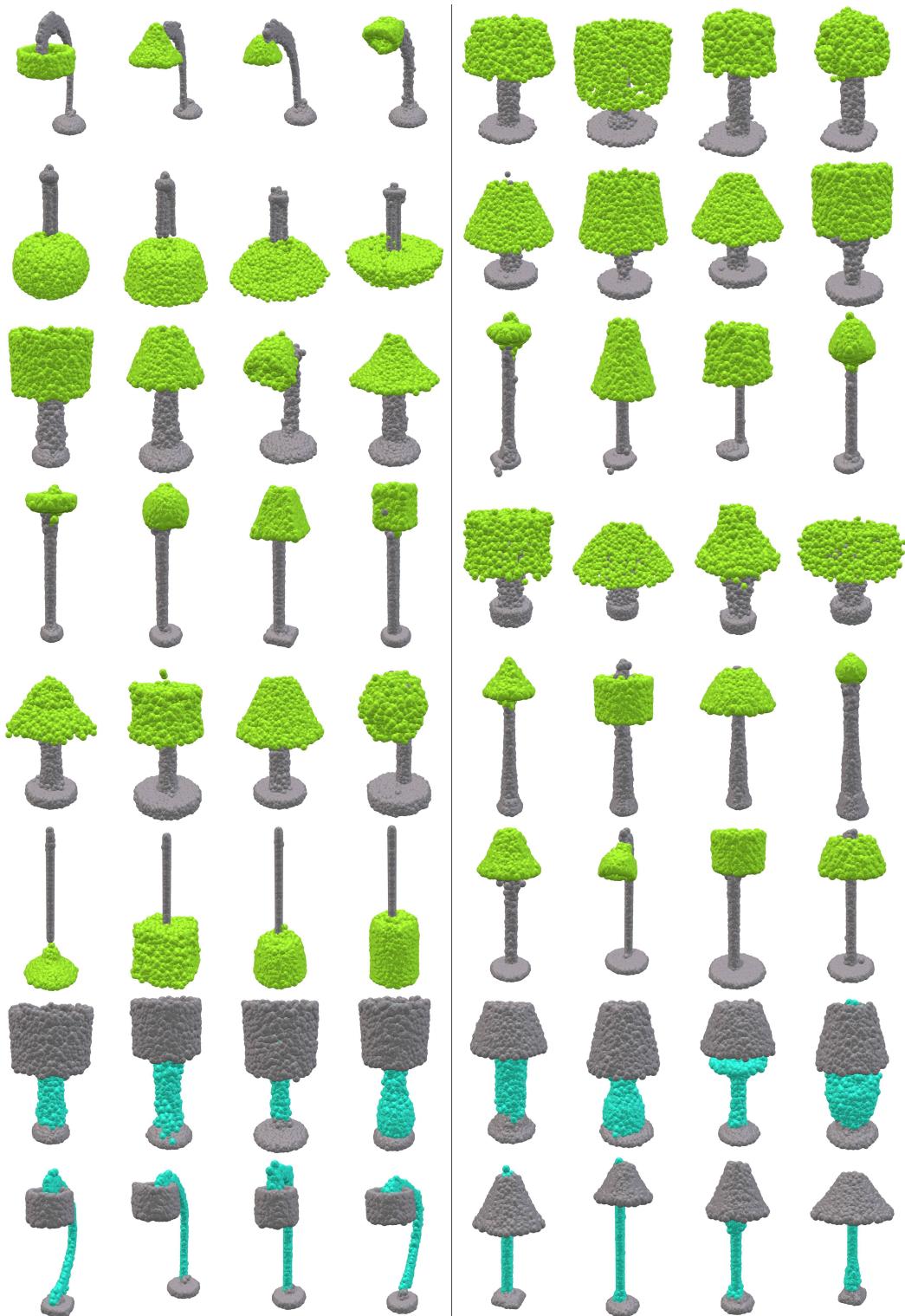
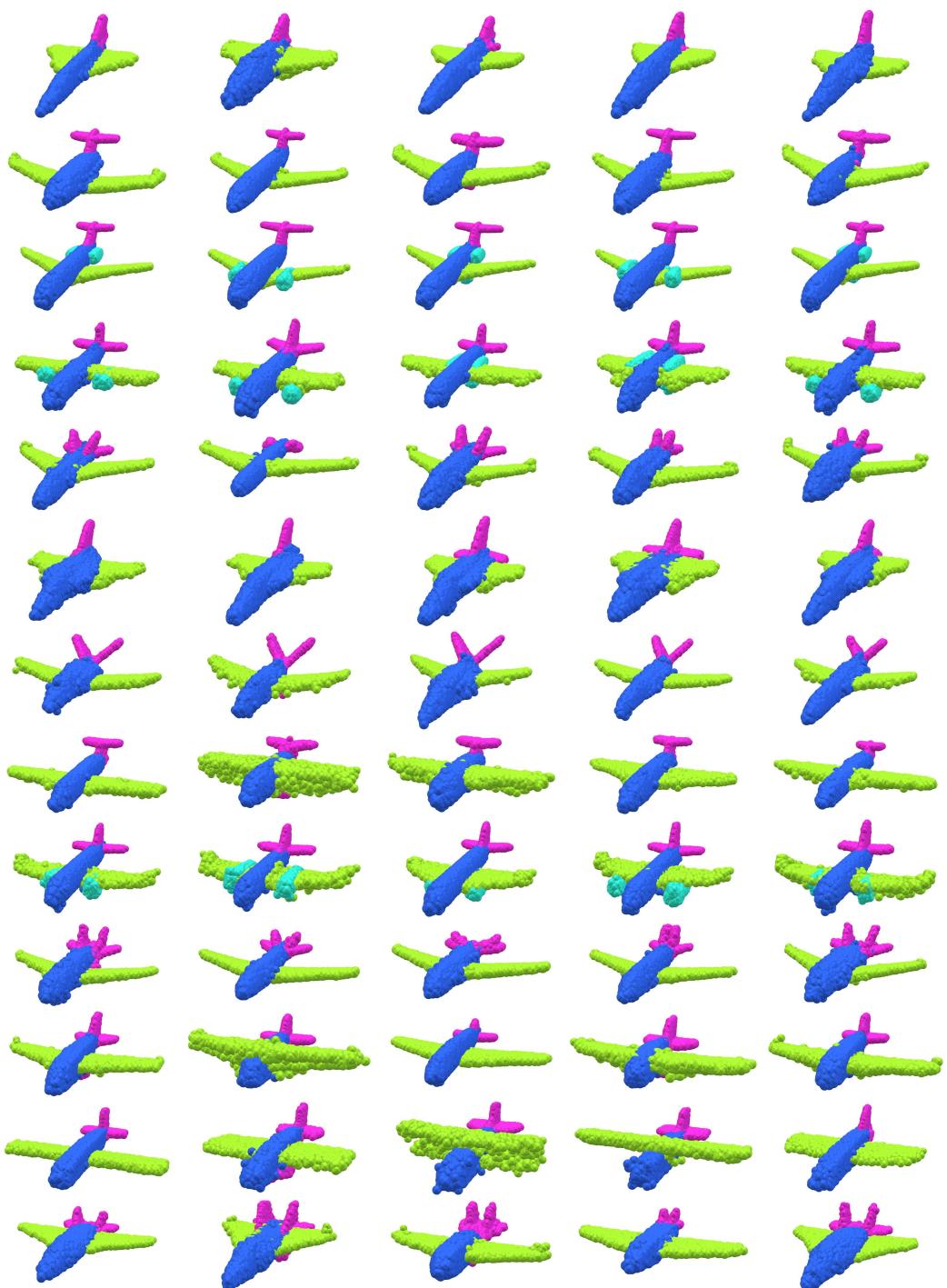


Figure S19: **Part-level Sampling: Lamps, Sampling one part.** Each row shows two sets of lamps (separated by the solid line) generated by sampling the colored part style latent while keeping the other part style latents (colored in grey) unchanged. Notice that the generated samples show diversity in style for colored parts, while the grey part's style stays fixed while changing its size and location to produce coherent shapes.



Figure S20: **Part-level Sampling: Lamps, Fixing one part.** Each row shows two sets of lamps (separated by the solid line) generated by sampling the set of part style latents associated with the colored parts while keeping the part style latent for the grey part fixed. Notice that the size and location of the grey part change accordingly to fit the different samples of part styles of the other parts



**Figure S21: Part Transformation Sampling: Airplanes.** Each row shows generated airplanes conditioned with different part transformation samples from the transformation sampler. All the part style latent for each row is kept fixed. Notice that the multimodality modeled by cIMLE training method allows for a diverse set of part transformation samples from the transformation sampler.

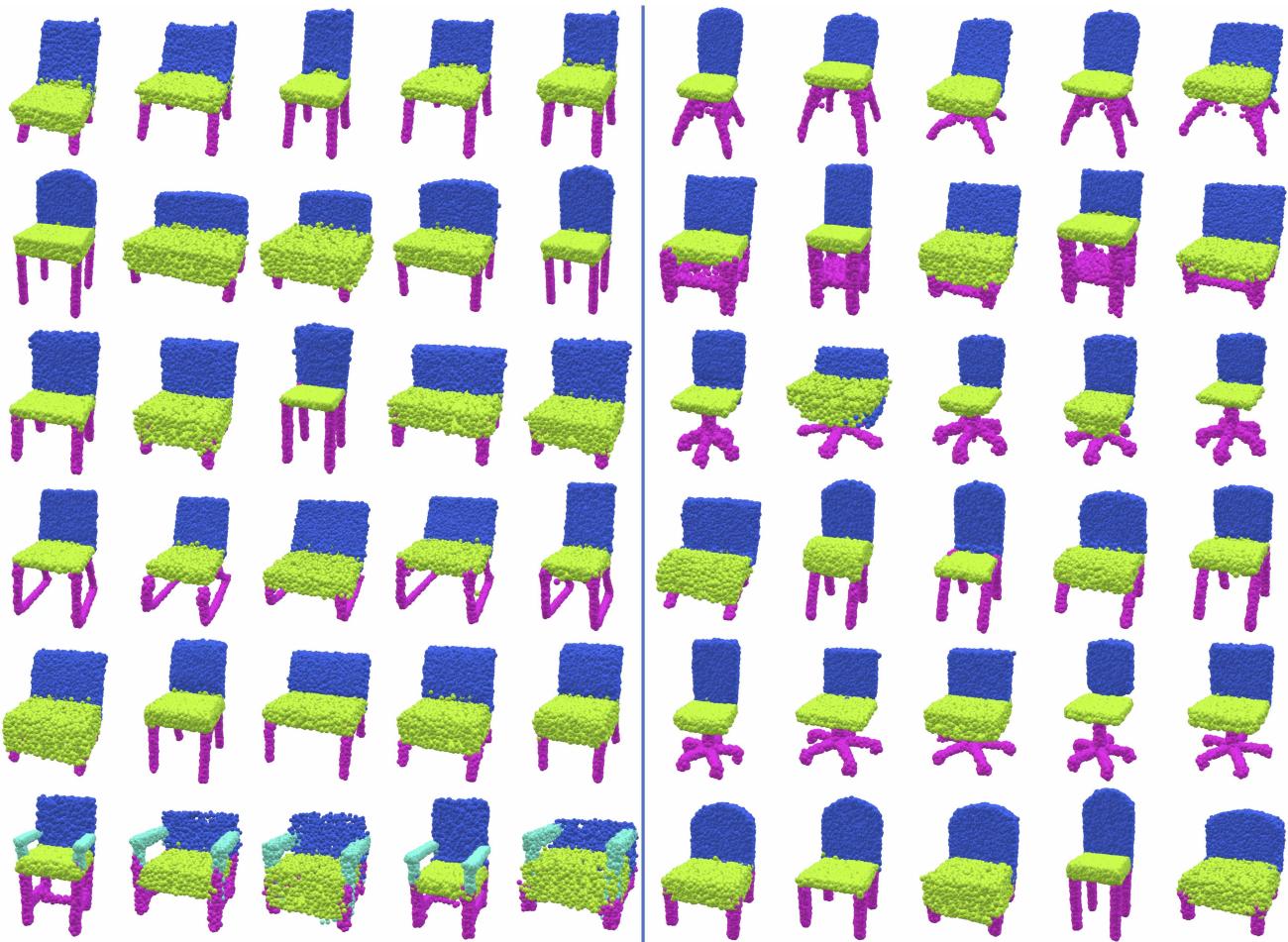


Figure S22: **Part Transformation Sampling: Chairs.** Each row shows generated airplanes conditioned with different part transformation samples from the transformation sampler. All the part style latent for each row is kept fixed.

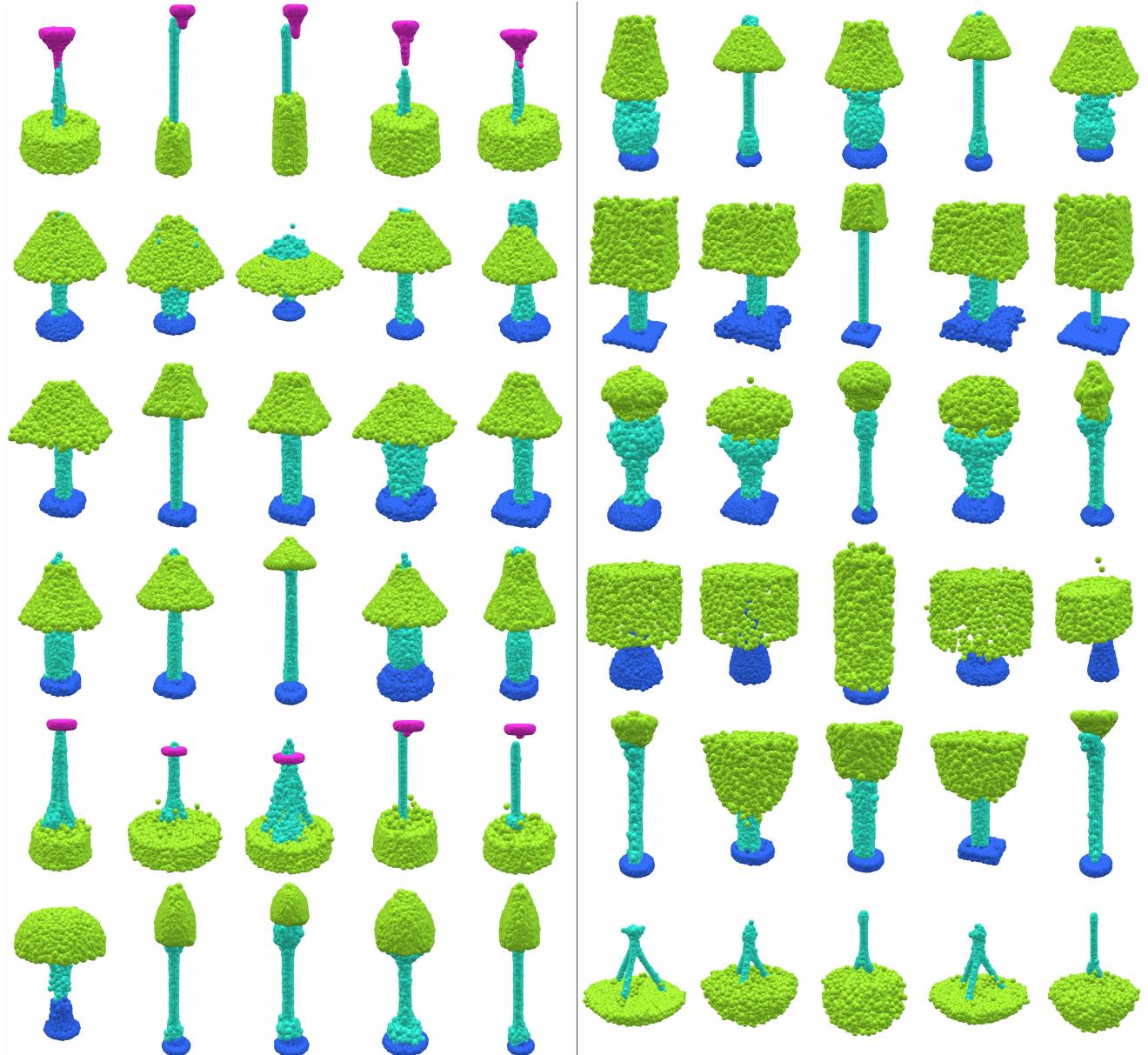


Figure S23: **Part Transformation Sampling: Lamps.** Each row shows generated airplanes conditioned with different part transformation samples from the transformation sampler. All the part style latent for each row is kept fixed.

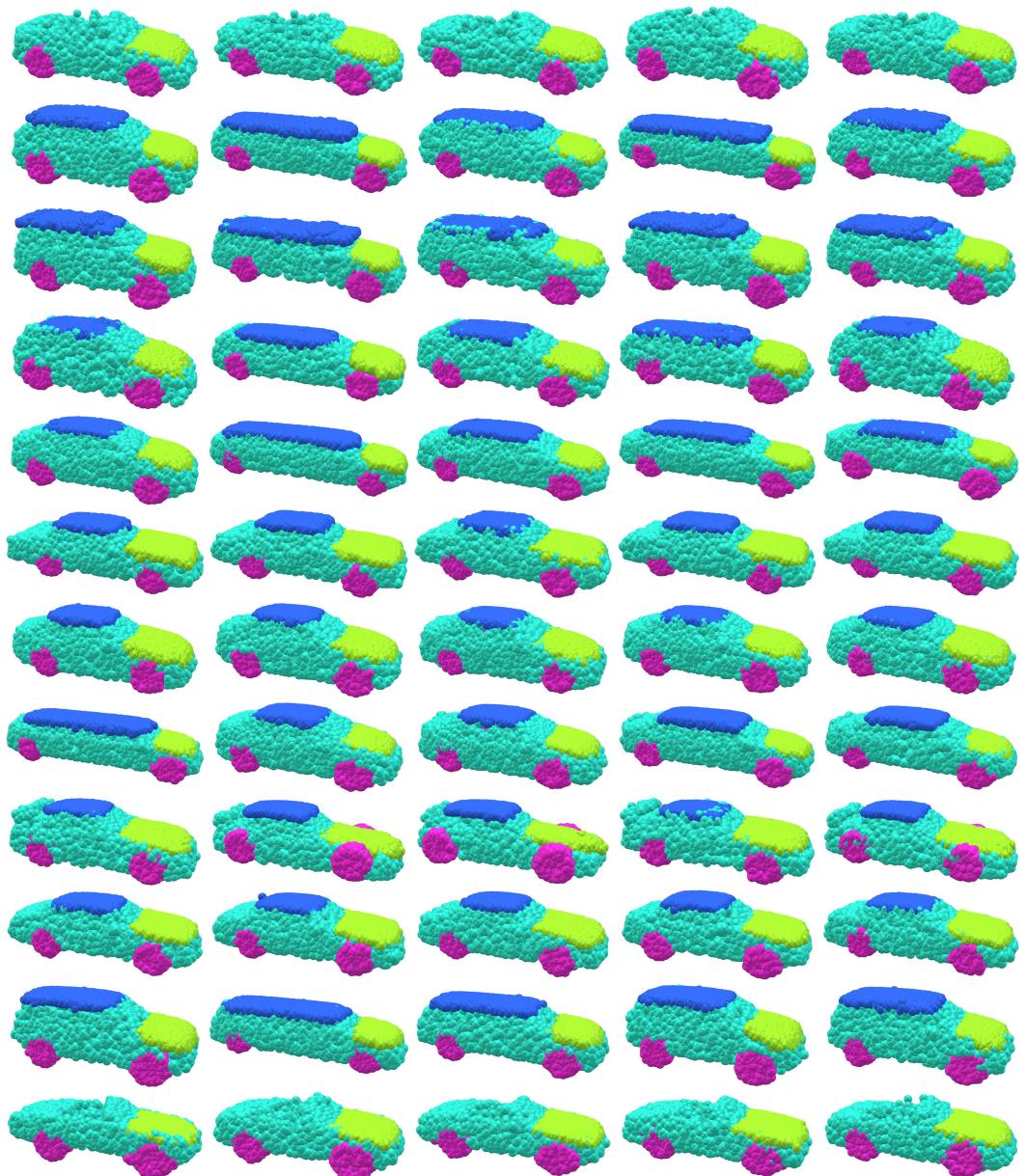


Figure S24: **Part Transformation Sampling: Cars.** Each row shows generated airplanes conditioned with different part transformation samples from the transformation sampler. All the part style latent for each row is kept fixed.

## S4. Network Details

We elaborate on each of our components in this section. Sec. S4.1 presents our part stylizers that learn an independent part style latent space for each part with a continuous normalizing flow model (CNF). Sec. S4.2 elaborates on our transformation sampler that is trained with conditional Maximum Likelihood Estimation (cIMLE) to model multimodality natural to the distribution of valid part configurations. Lastly, Sec. S4.3 details the Cross Diffusion Network using Denoising Diffusion Probabilistic Model (DDPM) with the generalized forward kernel.

### S4.1. Part Style Sampler

To model the part style distribution  $P_{\psi_j}(Z_j)$  for each part, we learn a variational encoder  $Q_{\varphi_j}(Z_j|\hat{S}_j)$  that models a Gaussian distribution with a learnable mean and diagonal variance given a canonicalized part  $\hat{S}_j$ . In practice, each part is canonicalized such that  $\hat{S}_j$  shifted by the mean and scaled by one standard deviation on each of the axes.

To efficiently learn the variation encoder, we use the reparametrization trick [25] giving us  $z_j = \mu_{\varphi_j} + \sigma_{\varphi_j} \varepsilon$  where  $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ .

Although it is possible to use a standard Gaussian for the prior distribution  $P_{\psi_j}(Z_j)$ , it has been shown [53, 34, 10] that a simple prior distribution used in VAE models can be less than ideal. To enrich the expressivity of the prior distribution, continuous normalizing flow (CNF) [53] can be used to parameterize the prior distribution as a learnable, invertible network. To this end, we parameterize each per part style distribution  $P_{\psi_j}(Z_j)$  as an invertible network and write the KL divergence as [53]

$$\text{KL}\left(P_{\psi_j}(z_j) \| Q_{\varphi_j}(z_j|\hat{S}_j)\right) = -\mathbb{E}_{z_j \sim Q_{\varphi_j}(Z_j|\hat{S}_j)} [\log P_{\psi_j}(z_j)] - H\left(Q_{\varphi_j}(Z_j|\hat{S}_j)\right). \quad (15)$$

for each part  $j = 1, \dots, m$  and for each shape  $S$ . Here  $H$  is the entropy and  $P_{\psi_j}(Z_j)$  is the learnable part style prior distribution obtained by transforming a standard Gaussian with an invertible network:

$$z_j := F_{\psi_j}(\xi(t_0)) = \xi(t_0) + \int_{t_0}^{t_1} f_{\psi_j}(\xi(t), t) dt$$

where  $\xi(t_0) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  and  $f_{\psi_j}$  is the continuous-time dynamics of the flow  $F_{\psi_j}$ . Then, the log probability can be calculated by

$$\log P_{\psi_j}(z_i) = \log P\left(F_{\psi_j}^{-1}(z_j)\right) + \int_{t_0}^{t_1} \text{tr}\left(\frac{\partial f_{\psi_j}}{\partial \xi(t)}\right) dt.$$

We finally note that for shapes with missing parts, we replace its part style latent with a dummy latent variance, and mask it out at training time.

### S4.2. Transformation Sampler

As detailed in Sec. 4.2 our the main paper, we train an implicit probabilistic model with conditional Implicit Maximum Likelihood (cIMLE) method. Specifically, we model a multimodal distribution of part transformations  $P_{\theta}(\mathbf{T}|\mathbf{z})$  given the part style latents by a deep neural network  $T_{\theta} : (\mathbf{z}, y) \mapsto \boldsymbol{\tau}$  where  $\mathbf{z} \sim P_{\psi}(\mathbf{Z})$  and  $y \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ , which is a 32-dimensional noise latent code. By sampling different noise latent code from the standard Gaussian, the neural network  $T_{\theta}$  is able to produce different valid modes of part transformations for the given set of part style latents. To train the implicit model with cIMLE, we follow [29] and cache the best fitting noise latent code for each set of part style latents  $\mathbf{z}$  during training, and optimize the transformation sampler using the cached noise codes for a number of iterations. The best fitting is defined by the cIMLE objective

$$\ell_{\boldsymbol{\tau}} = \sum_{S \in \mathbf{S}} \min_{k=1, \dots, K} \ell_{\text{fit}}(T_{\theta}(\mathbf{z}_S, y_k), \boldsymbol{\tau}_S), \quad (16)$$

where  $\boldsymbol{\tau}_S$  is the observed part transformations for shape  $S$ .  $\ell_{\text{fit}}$  is defined to be the distance between the generated transformation  $\{\boldsymbol{\tau}_k\}$  and observation  $\boldsymbol{\tau}_S$  summed across all parts:

$$\ell_{\text{fit}}(\boldsymbol{\tau}, \boldsymbol{\tau}_S) = \sum_{j=1}^m \|c_j - c_{S,j}\|_2^2 + \|\log s_j - \log s_{S,j}\|_2^2.$$

Same as in the main paper,  $c_j, c_{S,j} \in \mathbb{R}^3, s_j, s_{S,j} \in \mathbb{R}^3$  are the shifts and scales associated with  $j$ -th part of the generated transformation and the observed transformation respectively.

### S4.3. Cross Diffusion Network

To learn the data distribution  $P_\phi(S|\mathbf{z}, \boldsymbol{\tau})$ , we learn a point distribution  $P_\phi(X^{(0)}|\mathbf{z}, \boldsymbol{\tau}, j)$  for each part conditioned on the part style latents  $\mathbf{z}$  and the part transformations  $\boldsymbol{\tau}$ . For each observed shape  $S$ , represented as a point cloud, we assume that points from each part are sampled independently from  $P_\phi(X|\mathbf{z}, \boldsymbol{\tau}, j)$ . We define a forward diffusion process  $Q(X^{(0:T)}|\boldsymbol{\tau}_j)$  for each part using the generalized forward kernel with  $\mu = c_j$  and  $\Sigma = \text{Diag}(s_j)$  for each part. To learn the denoising process, we define a reverse process as another Markov chain  $P_\phi(X^{(0:T)}|\mathbf{z}, \boldsymbol{\tau}, j)$  for each part such that

$$P_\phi(X^{(0:T)}|\mathbf{z}, \boldsymbol{\tau}, j) = P(X^{(T)}|\boldsymbol{\tau}_j) \prod_{t=1}^T P_\phi(X^{(t-1)}|X^{(t)}, \mathbf{z}, \boldsymbol{\tau}, j)$$

where

$$P(X^{(T)}|\boldsymbol{\tau}_j) = \mathcal{N}(c_j, \text{Diag } s_j)$$

and each reversion kernel is parameterized as a Gaussian Distribution with learnable means to approximate the posterior distribution  $q(X^{(t-1)}|X^{(t)}, X^{(0)}, c_j, \text{Diag}(s_j))$ :

$$P_\phi(X^{(t-1)}|X^{(t)}, \mathbf{z}, \boldsymbol{\tau}, j) = \mathcal{N}\left(\Xi_\phi\left(t, X^{(t)}, \mathbf{z}, \boldsymbol{\tau}, j\right), \eta_t^2 \text{Diag}(s_j)\right).$$

Following [34, 60, 39, 59], instead of directly learning the mean  $\Xi_\phi$ , we reparameterize the forward posterior distribution according to Eq. 14 to learn  $\varepsilon_\phi$  that approximate noises from standard Gaussian. Thus, the final objective becomes

$$\ell_{\text{cross}} = \sum_{j=1}^m \sum_{x \in S_j} \mathbb{E}_{\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \mathbf{z} \sim Q_\varphi, t \in \{1, \dots, T\}} \left[ \left\| \varepsilon - \varepsilon_\phi(x^{(t)}, \mathbf{z}, \boldsymbol{\tau}, j, t) \right\|_2^2 \right]. \quad (17)$$

## S5. Implementation Details

In this section, we provide the implementation details: our training pipeline and hyperparameters, network architecture, and training and sampling algorithms.

### S5.1. Training Details

As detailed in the main paper, our training loss is transformed to  $\ell_{\text{total}} = \ell_{\text{recon}} + \lambda_1 \ell_{\mathbf{Z}} + \lambda_2 \ell_{\boldsymbol{\tau}}$ . In practice, we adopt a two-stage training strategy. During the first stage, we train the prior flow  $P_\psi(Z_j)$ , variational encoders  $Q_\varphi(\mathbf{Z}|\hat{S})$ , and the cross diffusion network  $P_\phi(X|\mathbf{Z}, \boldsymbol{\tau}_S, j)$ . For the Cross Diffusion Network, We use timestep  $T = 100$  for all of our models and the variance scheduling parameters  $\alpha_t$  are set to linearly decrease from 0.9999 to 0.08 following the setting of [34]. Ground truth transformations are used to condition the Cross Diffusion network during the first stage. The loss optimized for the first stage of training is

$$\begin{aligned} \ell_{\text{first stage}} &= \lambda_1 \ell_{\mathbf{Z}} + \ell_{\text{recon}} \\ &= \mathbb{E}_{S \sim P(S)} \left[ -\lambda_1 \mathbb{E}_{z_j \sim Q_{\varphi_j}(Z_j|\hat{S}_j)} [\log P_{\psi_j}(z_j)] - H(Q_{\varphi_j}(Z_j|\hat{S}_j)) \right. \\ &\quad \left. + \frac{1}{m} \sum_{j=1}^m \frac{1}{|S_j|} \sum_{x \in S_j} \mathbb{E}_{\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \mathbf{z} \sim Q_\varphi, t \in \{1, \dots, T\}} \left[ \left\| \varepsilon - \varepsilon_\phi(x^{(t)}, \mathbf{z}, \boldsymbol{\tau}_S, j, t) \right\|_2^2 \right] \right]. \end{aligned} \quad (18)$$

where  $\lambda_1 = 5e-4$  for chairs and  $\lambda_1 = 1e-3$  for lamps, airplanes, and cars. For all models, we use a batch size of 128, and the first stage is trained for 8000 epochs using Adam [24] optimizer without weight decay. Momentum parameters  $\beta_1$  and  $\beta_2$  for Adam optimizer are set to 0.9 and 0.999 respectively. The learning rate is set to 2e-3 at the start of training as is linearly decreased starting from 4000 epochs to 1e-4 at the end of training. The first stage takes around 36 hours for chairs, 30 hours for airplanes, 24 hours for lamps, and 16 hours for cars.

In the second stage, we freeze the weights of the part stylizers and the cross diffusion network, and only train the transformation sampler using the cIMLE training strategy. The second stage loss is the cIMLE loss plus the reconstruction loss:

$$\ell_{\text{second stage}} = \ell_{\text{recon}} + \lambda_2 \ell_{\boldsymbol{\tau}}, \quad (19)$$

where  $\lambda_2 = 1$  for all classes. The training of cIMLE requires recaching of noise latent codes. Specifically, we recache the noise latent codes for each shape in our training data every 50 epoch, during which we sample  $K = 20$  noise latent code and associate each set of part style latents  $z$  with a best fitting code according to Eq. 16. These pairs are then trained for 50 epochs before the noise codes are recached. The second stage is trained for a maximum of 4000 epochs and the models used for evaluation are selected from the best checkpoint. Similar to the first stage, we use the Adam optimizer, and keep the learning rate fixed at 2e-4 for the entire training process.

## S5.2. Network Parameters

### S5.2.1 Part Style Sampler

Each variation encoder  $Q_{\varphi_j}(Z_j | \hat{S}_j)$  is implemented as a PointNet [7] following the architecture of [34, 5] with a shared per point feature regression layer. Specifically, we feed the canonicalized parts through a 3-128-256-512 MLP layer with ReLU nonlinearity and batch normalization. We share the per-point MLP across all parts to reduce parameter count. The per-point features are then max-pooled for each part to obtain a 512 dimensional feature for each part. Then, the feature is fed into a 512-256-128-256 MLP with the ReLU nonlinearity to output a 256 dimensional mean and a diagonal variance which parameterize the Gaussian distribution of  $Q_{\varphi_j}(Z_j | \hat{S}_j) = \mathcal{N}(\mu_{\varphi_j}(\hat{S}_j), \sigma_{\varphi_j}(\hat{S}_j))$ . We note that the MLP after max pooling is not shared across different parts as the objective decomposition assumes independence of part style latent distributions. The 256 dimensional part style latent is then sampled from this Gaussian for downstream training. Because each part style latent distribution is independent of the other, we use one prior flow network for each part. Each prior flow is implemented with the same architecture as [34, 53]. Specifically, we use 14 affine coupling layers with the dimension of hidden states being 256, identical to the dimension of part style latents. Following each of the layers, we apply moving batch normalization [22, 15]. Both the scaling and translation networks  $F(\cdot)$  and  $G(\cdot)$  are 128-256-256-128 MLPs with ReLU nonlinearity.

### S5.2.2 Transformation Sampler

The network for transformation sampler  $T_\theta$  is implemented using a self-attention-only transformer following [39, 42]. The input is  $m$  tokens with each token being the concatenation of the part style latent  $z_j \in \mathbb{R}^{256}$  and a scaled noise latent code  $\lambda y \in \mathbb{R}^{32}$ . The  $\lambda$  parameter is class specific and controls the degree of transformation variations produced by different modes of the transformation sampler. For the reported models, we use  $\lambda = 100$  for chairs,  $\lambda = 50$  for airplanes and cars, and  $\lambda = 10$  for lamps. Then, each token is projected back to a 256 dimensional vector and the part label associated with each token is added as a learnable embedding vector to the projected token. For the transformer architecture, we use 5 layers of multi-head self-attention followed by a feed-forward layer and layer normalization. Parameters can be found in Tab. S4. The output of  $T_\theta$

parameters	Values
layers	5
head dimension	32
num heads	8
drop out rate	0

Table S4: Parameters for Transformation Sampler.

is a three-dimensional scale and shift vector for each part style latent. For missing parts, we mask out its associated token during the self-attention layers.

### S5.2.3 Cross Diffusion Network

As derived in Eq (14), we learn a noise approximator  $\varepsilon_\phi$  that estimates pure noises given noisy inputs at time step  $t$ . We use a cross-attention-only transformer to implement  $\varepsilon_\phi$ . Specifically, the input consists of  $N$  tokens where  $N$  is the number of points in a shape  $S$ . Each input token is the concatenation of  $(x^{(t)}, \tau_j, j)$  if the point  $x^{(t)}$  belongs to the  $j$ -th part. Because of computation cost and independence assumption across different point samples, the input tokens are first projected to 128 dimensional vectors and then individually attend to  $m$  context tokens each being the concatenation of  $(z_j, \tau_j, j, t)$ , for  $j = 1, \dots, m$  followed by a feedforward layer and layer normalization. The transformer architecture is similar to that of the transformation sampler, with parameters listed in Tab. S5.

parameters	Values
layers	5
head dimension	16
num heads	8
drop out rate	0.2

Table S5: Parameters for Cross Diffusion Network.

### S5.3. Algorithm

We provide the training and sampling algorithms for DiffFacto in Algorithm 1 (first stage training), Algorithm 2 (second stage training), and Algorithm 3 (sampling).

---

#### Algorithm 1 DiffFacto Training: Stage 1

---

```

repeat
    Let  $S \in \mathcal{S}_{\text{data}}$ 
    for  $j = 1, \dots, m$  do
        if  $S_j \neq \emptyset$  then
            Sample  $z_j \sim Q_{\varphi_j} (Z_j | \hat{S}_j)$ .
        else
             $z_j = z_{\text{dummy}}$ .
        end if
    end for
     $z \leftarrow \{z_j\}_{j=1}^m$ 
    Let  $\tau_S$  to be the part transformations for  $S$ .
    Sample  $t \sim \text{Uniform} (\{1, \dots, T\})$ 
    Sample  $\varepsilon \sim \mathcal{N} (\mathbf{0}, \mathbf{I})$ 
    Compute  $\nabla_{\phi, \varphi, \psi} [\ell_{\text{first stage}}]$ ; then perform gradient descent.
until converged.

```

---

## S6. Experiment Set-up Additional Details

### S6.1. Control-Enabled Baselines Additional Details

Because none of the existing networks enable part-level generation, we modify two state-of-the-art baselines (LION [59] and ShapeGF [5]) to enable part-level sampling for a fair comparison with DiffFacto, we refer to these control-enabled modification as Ctrl-LION and Ctrl-ShapeGF. To enable part-level generation, we train their method per part, *i.e.* one model for each part of an object class. We use the same dataset as our method, namely ShapeNet with semantic segmentation labels.

As a result, we obtain a generative model for each part for both ShapeGF and LION networks. Then, to obtain the global shape, we simply sample from each of the perpart Ctrl-ShapeGF and Ctrl-LION and concatenate the points together. To sample on the part level, we simply sample from a specific part latent space while fixing the rest of the generated shapes.

### S6.2. Evaluation Protocol Additional Details

We elaborate on the evaluation protocol we use to compare with baselines here. Sec. S6.2.1 details the intro-part evaluation procedure, which we use to evaluate the quality and diversity of our per-part distributions. Sec. S6.2.2 elaborate on the snapping metric, which we use to measure against the control-enabled baselines. Sec. S6.2.3 details the human study that was conducted.

#### S6.2.1 Intra-part Evaluation Additional Details

As presented in the main paper, we use the standard generation metrics to measure the similarity between the distributions of canonicalized parts of the generated shapes compared to the test set from [56] of segmented shapes. For a set of canonicalized part  $S_{t,j}$  from the test set and a set of canonicalized part  $S_{g,j}$  from the generated set, the generation metrics are computed as

---

**Algorithm 2** DiffFacto Training: Stage 2

---

```

repeat
  for  $S \in S_{\text{data}}$  do
    Let  $\tau_S$  to be the part transformations for  $S$ .
    for  $j = 1, \dots, m$  do
      if  $S_j \neq \emptyset$  then
        Sample  $z_j \sim Q_{\varphi_j}(Z_j | \hat{S}_j)$ .
      else
         $z_j = z_{\text{dummy}}$ .
      end if
    end for
     $z \leftarrow \{z_j\}_{j=1}^m$ 
    Sample  $y_1, \dots, y_K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
    Set  $y_S^* = \arg \min_{y_i \in \{y_1, \dots, y_K\}} \ell_{\text{fit}}(T_\theta(z, y_i), \tau_S)$ .
  end for
  for  $l = 1, \dots, 50$  do
    Sample  $S \in S_{\text{data}}$ 
    for  $j = 1, \dots, m$  do
      if  $S_j \neq \emptyset$  then
        Sample  $z_j \sim Q_{\varphi_j}(Z_j | \hat{S}_j)$ .
      else
         $z_j = z_{\text{dummy}}$ .
      end if
    end for
     $z \leftarrow \{z_j\}_{j=1}^m$ 
    Let  $\tau_S$  to be the part transformations for  $S$ .
    Sample  $t \sim \text{Uniform}(\{1, \dots, T\})$ 
    Sample  $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
    Compute  $\nabla_\theta [\ell_{\text{recon}} + \ell_{\text{fit}}(T_\theta(z, y_S^*), \tau_S)]$ ; then perform gradient descent.
  end for
until converged.

```

---

### Minimum matching distance (MMD-P)

$$\text{MMD-P}(S_{g,j}, S_{t,j}) = \frac{1}{|S_{t,j}|} \sum_{S_{t,j} \in S_{t,j}} \min_{S_{g,j} \in S_{g,j}} \text{Chamfer}(S_{g,j}, S_{t,j}).$$

The idea behind MMD-P is to calculate the average distance between the point clouds in the reference set and their closest neighbors in the generated set. A smaller MMD-P implies that the parts from the test set are well represented in the parts from the generated set.

### Coverage (COV-P)

$$\text{COV-P}(S_{g,j}, S_{t,j}) = \frac{|\{\arg \min_{S_{t,j} \in S_{t,j}} \text{Chamfer}(S_{t,j}, S_{g,j})\}|}{|S_{t,j}|}.$$

A higher COV-P percentage means that parts in the test set are well covered by the parts in the generated set, so it implies that the generated parts are diverse.

### 1-NN classifier accuracy (1NNA-P)

$$\text{1NNA-P}(S_{g,j}, S_{t,j}) = \frac{\sum_{S_{g,j} \in S_{g,j}} \mathbb{1}[N_{S_{g,j}} \in S_{g,j}] + \sum_{S_{t,j} \in S_{t,j}} \mathbb{1}[N_{S_{t,j}} \in S_{t,j}]}{|S_{t,j}| + |S_{g,j}|}.$$

where  $\mathbb{1}[\cdot]$  is the indicator function and  $N_{S_j}$  is the nearest neighbor of  $S_j$  in the set  $S_{t,j} \cup S_{g,j} \setminus \{S_j\}$  (i.e., the union of the sets  $S_{g,j}$  and  $S_{t,j}$  excluding,  $S_j$ ). With this definition, 1NNA-P represents the leave-one-out accuracy of the 1-NN classifier

---

**Algorithm 3** DiffFacto Sampling

---

```

for  $j = 1, \dots, m$  do
    Sample  $\xi \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ .
     $z_j = F_{\psi_j}(\xi(t_0))$ 
end for
 $z \leftarrow \{z_j\}_{j=1}^m$ 
Sample  $y \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
Compute  $\tau = \{(c_j, s_j)\}_{j=1}^m = T_\theta(z, y)$ 
for  $j = 1, \dots, m$  do
     $S_j^{(T)} = \{x^{(T)}\} \sim \mathcal{N}(c_j, \text{Diag}(s_i))$ 
end for
 $S^{(T)} \leftarrow \{S_j^{(T)}\}_{j=1}^m$ 
for  $t = T, \dots, 1$  do
    for  $i = 1, \dots, m$  do
        for  $x^{(t)} \in S_j^{(t)}$  do
             $x^{(t-1)} \sim P_\theta(X^{(t-1)} | x^{(t)}, z, \tau, j)$ 
        end for
         $S_i^{(t-1)} \leftarrow \{x^{(t-1)}\}$ 
    end for
     $S^{(t-1)} \leftarrow \{S_j^{(t-1)}\}_{j=1}^m$ 
end for
return  $S^{(0)}$ 

```

---

(measured in Chamfer distance) defined above. If the performance of the 1NN classifier is close to 50 percent, the parts from the test set and the generated set are well mingled. Thus, 1NNA-P directly measures the similarity between the test parts and the generated parts, both in diversity and quality.

To prepare for the sets of canonicalized parts, we use 512 points for each of the part samples for the test set. For our model and all the baselines, we sample 2048 points on the entire shape and obtain for each shape its canonicalized parts from these points. We sample the same number of shapes as in the test set. Since the baselines do not generate shapes with part segmentation, we enable their evaluation using a segmentor with PointNet++ backbone [40] trained on the ShapeNetCore PartSeg dataset. Then, MMD-P, COV-P, and 1NNA-P is computed for each part and in the end, the score is reported as a weighted average based on the number of part samples in each of the part sets of the test shapes. For implicit methods, we sample 2048 points from their generated mesh. The canonicalization of each part is done by fitting all three axes into the unit cube.

### S6.2.2 Inter-Part Evaluation Additional Details

To measure the connectivity of the generated shapes, we select a set of connected parts and compute the Chamfer distance between the closest  $N_{\text{SNAP}} = 30$  points between the two parts. Specifically, if the  $j$ -th part is connected to a set of parts  $\{S_k\}$  (note that a part can be connected to multiple parts due to different structures of shapes), we define the snapping distance of connection  $j$  to be

$$\text{SNAP}(S_j) = \min_{S_k \in \{S_k\}} \text{Chamfer}\left(N_{S_k}^{(N_{\text{SNAP}})}(S_j), N_{S_j}^{(N_{\text{SNAP}})}(S_k)\right),$$

where  $N_X^{(N_{\text{SNAP}})}(Y)$  are the  $N_{\text{SNAP}}$  closest points in shape  $Y$  to shape  $X$ .

For chairs, we define connections of the back, seat, and arms respectively. The back connection is defined to *connect to either the legs or the seat*; the seat connection is defined to be *connected to the legs*, and lastly, the arm connection is defined to *connect to either the back or the seat*.

### S6.2.3 Human Study Additional Details

We provide further details on the user study we conducted to evaluate controllability of our method compared to the control-enabled baselines (Ctrl-ShapeGF and Ctrl-LION). A user is asked to select the triplet of edited shapes that are most

plausible, where examples of plausible shapes to be: 1) does not have detachment of parts, 2) does not have interpenetrating parts, 3) the size of the edited parts agrees with the rest of the shape, which was included in the prompt. As described in the main paper, a (controlled) edit is defined as re-sampling a pre-selected part of the given shape. Our user study contained 10 questions, which are randomly selected shapes from the dataset, each with random parts to edit. Fig. S25 shows one example question from our user study.

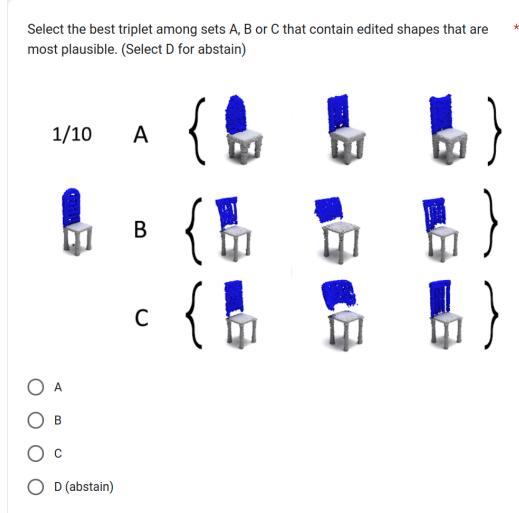


Figure S25: **User Study Example.** The subject is asked ten questions similar to the example above in which they are asked to pick the most coherent group out of the three.

## References

- [1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49. PMLR, 2018. [2](#)
- [2] Panos Achlioptas, Judy Fan, Robert X. D. Hawkins, Noah D. Goodman, and Leonidas J. Guibas. Shapeplot: Learning language for shape differentiation. *CoRR*, abs/1905.02925, 2019. [9](#)
- [3] Péter Borosán, Reid Howard, Shaoting Zhang, and Andrew Nealen. Hybrid mesh editing. In *Eurographics (short papers)*, pages 41–44, 2010. [2](#)
- [4] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015. [2](#)
- [5] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *ECCV*, 2020. [1, 2, 3, 6, 7, 14, 15, 41, 42](#)
- [6] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. [6](#)
- [7] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017. [41](#)
- [8] Siddhartha Chaudhuri, Daniel Ritchie, Jiajun Wu, Kai Xu, and Hao Zhang. Learning generative models of 3d structures. In *Computer Graphics Forum*, volume 39, pages 643–666. Wiley Online Library, 2020. [2](#)
- [9] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018. [4](#)
- [10] Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder, 2016. [39](#)
- [11] Fabrizio Corda, Jean-Marc Thiery, Marco Livesu, Enrico Puppo, Tamy Boubekeur, and Riccardo Scateni. Real-time deformation with coupled cages and skeletons. In *Computer Graphics Forum*, volume 39, pages 19–32. Wiley Online Library, 2020. [2](#)
- [12] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. [2](#)

- [13] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao(Richard) Zhang. SDM-NET: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2019)*, 38(6):243:1–243:15, 2019. 2
- [14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 2
- [15] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models, 2018. 41
- [16] Zekun Hao, Hadar Averbuch-Elor, Noah Snavely, and Serge Belongie. Dualsdf: Semantic shape manipulation using a two-level representation. pages 7631–7641, 2020. 2
- [17] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. Spaghetti: Editing implicit shapes through part aware generation. *ACM Transactions on Graphics (TOG)*, 41(4):1–20, 2022. 2
- [18] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. Spaghetti: Editing implicit shapes through part aware generation. In *SIGGRAPH Asia*, 2022. 3, 7, 13
- [19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 2, 5
- [20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arxiv:2006.11239*, 2020. 13
- [21] Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3d shape generation, 2022. 2, 13
- [22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 41
- [23] Hyeongju Kim, Hyeonseung Lee, Woo Hyun Kang, Joun Yeop Lee, and Nam Soo Kim. Softflow: Probabilistic framework for normalizing flow on manifolds. *Advances in Neural Information Processing Systems*, 33:16388–16397, 2020. 2
- [24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. 40
- [25] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. 2, 39
- [26] Juil Koo, Ian Huang, Panos Achlioptas, Leonidas J Guibas, and Minhyuk Sung. Partplot: Learning shape part segmentation from language reference games. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16505–16514, 2022. 2
- [27] Jun Li, Chengjie Niu, and Kai Xu. Learning part generation and assembly for structure-aware shape synthesis. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11362–11369, 2020. 2
- [28] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017. 2
- [29] Ke Li and Jitendra Malik. Implicit maximum likelihood estimation. *arXiv preprint arXiv:1809.09087*, 2018. 4, 39
- [30] Ke Li, Shichong Peng, Tianhao Zhang, and Jitendra Malik. Multimodal image synthesis with conditional implicit maximum likelihood estimation. *International Journal of Computer Vision*, 128:2607–2628, 2020. 2, 4, 8, 9
- [31] Ruihui Li, Xianzhi Li, Ka-Hei Hui, and Chi-Wing Fu. Sp-gan: Sphere-guided 3d shape generation and manipulation. *ACM Transactions on Graphics (TOG)*, 40(4):1–12, 2021. 2, 13
- [32] Shidi Li, Miaomiao Liu, and Christian Walder. Editvae: Unsupervised part-aware controllable 3d point cloud shape generation, 2021. 2
- [33] Connor Z. Lin, Niloy J. Mitra, Gordon Wetzstein, Leonidas Guibas, and Paul Guerrero. Neuform: Adaptive overfitting for neural shape editing, 2022. 2
- [34] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *CVPR*, June 2021. 1, 2, 3, 5, 6, 7, 14, 39, 40, 41
- [35] Zhaoyang Lyu, Zhifeng Kong, Xudong Xu, Liang Pan, and Dahua Lin. A conditional point diffusion-refinement paradigm for 3d point cloud completion. *arXiv preprint arXiv:2112.03530*, 2021. 2
- [36] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014. 2, 4, 8, 9
- [37] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 909–918, 2019. 3
- [38] Charlie Nash and Christopher KI Williams. The shape variational autoencoder: A deep generative model of part-segmented 3d objects. In *Computer Graphics Forum*, volume 36, pages 1–12. Wiley Online Library, 2017. 2
- [39] 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, 40, 41
- [40] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 7, 44
- [41] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015. 2
- [42] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 41
- [43] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 2

- [44] Thomas W Sederberg and Scott R Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160, 1986. 2
- [45] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015. 2, 4, 8, 9
- [46] 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*, pages 61–70, 2020. 2
- [47] Minhyuk Sung, Zhenyu Jiang, Panos Achlioptas, Niloy J Mitra, and Leonidas J Guibas. Deformsyncnet: Deformation transfer via synchronized shape deformation spaces. *arXiv preprint arXiv:2009.01456*, 2020. 2
- [48] Minhyuk Sung, Hao Su, Vladimir Kim, Siddhartha Chaudhuri, and Leonidas Guibas. ComplementMe: Weakly-supervised component suggestions for 3d modeling. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 2017. 1
- [49] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679, 2020. 2
- [50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2
- [51] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X Chang, and Daniel Ritchie. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics (TOG)*, 38(4):1–15, 2019. 2
- [52] Qun-Ce Xu, Tai-Jiang Mu, and Yongliang Yang. A survey of deep learning-based 3d shape generation. *Computational Visual Media*, 2022. 1
- [53] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *ICCV*, 2019. 1, 2, 3, 6, 7, 39, 41
- [54] Jie Yang, Kaichun Mo, Yu-Kun Lai, Leonidas J Guibas, and Lin Gao. Dsg-net: Learning disentangled structure and geometry for 3d shape generation. *ACM Transactions on Graphics (TOG)*, 42(1):1–17, 2022. 2
- [55] Ximing Yang, Yuan Wu, Kaiyi Zhang, and Cheng Jin. Cpcgan: A controllable 3d point cloud generative adversarial network with semantic label generating. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3154–3162, 2021. 2
- [56] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *SIGGRAPH Asia*, 2016. 6, 7, 42
- [57] Kangxue Yin, Zhiqin Chen, Siddhartha Chaudhuri, Matthew Fisher, Vladimir Kim, and Hao Zhang. Coalesce: Component assembly by learning to synthesize connections. In *3DV*, 2020. 1, 2
- [58] Shin Yoshizawa, Alexander G Belyaev, and Hans-Peter Seidel. Free-form skeleton-driven mesh deformations. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 247–253, 2003. 2
- [59] 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 *NeurIPS*, 2022. 1, 2, 6, 7, 14, 15, 40, 42
- [60] 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)*, pages 5826–5835, October 2021. 2, 5, 40
- [61] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through voxel diffusion. In *ICCV*, pages 5826–5835, October 2021. 2