GPLD3D: Latent Diffusion of 3D Shape Generative Models by Enforcing Geometric and Physical Priors

Yuan Dong*, Qi Zuo* , Xiaodong Gu, Weihao Yuan, Zhengyi Zhao, Zilong Dong, Liefeng Bo
Institute for Intelligent Computing, Alibaba Group
Hang Zhou, Zhejiang, China
{dy283090, muyuan.zq, dadong.gxd, qianmu.ywh, bushe.zzy, list.dzl, liefeng.bo}@alibaba-inc.com

Qixing Huang
The University of Texas at Austin
Austin, Texas, USA
huangqx@cs.utexas.edu

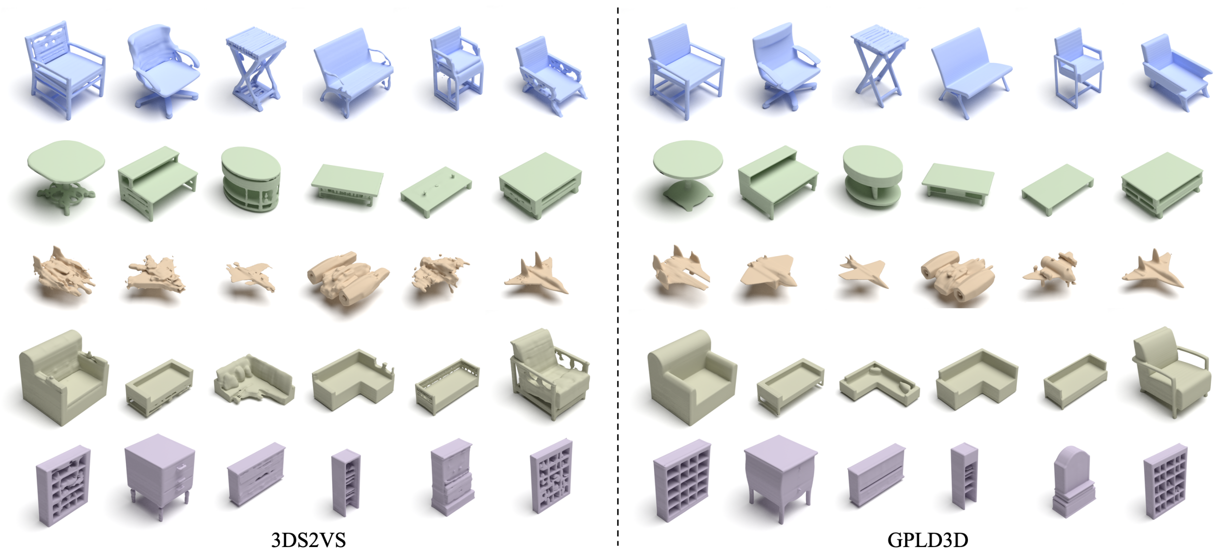


Figure 1. (Left) Synthetic shapes generated by 3DS2VS [56], which present various issues in geometric feasibility and physical stability. (Right) Synthetic shapes generated by GPLD3D, which have significantly improved geometric feasibility and physical stability.

Abstract

State-of-the-art man-made shape generative models usually adopt established generative models under a suitable implicit shape representation. A common theme is to perform distribution alignment, which does not explicitly model important shape priors. As a result, many synthetic shapes are not connected. Other synthetic shapes present problems of physical stability and geometric feasibility. This paper introduces a novel latent diffusion shape-generative model regularized by a quality checker that outputs a score of a latent code. The scoring function employs a learned function that provides a geometric feasibility score and a deterministic procedure to quantify a physical stability score. The key to our approach is a new diffusion procedure that combines the discrete empirical data distribution and a continuous distribution induced by the quality checker. We introduce a principled approach to determine the tradeoff parameters for learning the denoising network at different noise levels. Experimental results show that our approach outperforms state-of-the-art shape generations quantitatively and qualitatively on ShapeNet-v2.

1. Introduction

Thanks to recent advances in generative models such as generative adversarial networks (GAN) [4, 16, 58, 61], variational auto-encoders (VAE) [22, 44], normalizing flows (NF) [33, 53], autoregressive (AR) [52, 55], and denoising diffusion probabilistic models (DDPM) [39, 40], there is growing interest in adopting these generative models in the 3D domain by developing a suitable 3D representation, for example, volumetric grid [50], point cloud [3, 53], triangular mesh [24, 32], implicit representation [12, 28, 36, 36, 56]. However, a common theme of these generative models is to match the empirical distribution defined by the training data and the induced distribution derived from a prior distribution of the latent space. These approaches do not explicitly model rich shape properties in the 3D domain that are critical for downstream applications. Consider many state-of-the-art shape generators that use implicit shape representations. Synthetic shapes often have disconnected pieces, with additional issues of physical stability and geometric feasibility.

A major issue with existing techniques is that they only see the training instances, which are a very sparse set of samples. However, they do not model the geometrical and physical properties of synthetic instances. Such issues cannot be easily addressed by developing suitable neural representations. As man-made shapes have diverse topological structures, enforcing these properties under representations that can model varying topologies, e.g., implicit surfaces and point clouds, remains extremely challenging.

In this paper, we introduce a novel approach, called GPLD3D, which greatly enhances the geometrical feasibility and physical stability of synthetic shapes. We employ the latent diffusion paradigm [12, 34, 36, 56], which has proven to be a state-of-the-art shape-generative model. Consider a pre-trained generative model that maps the latent space to the shape space. Unlike training a diffusion model that maps the Gaussian distribution of the latent space to the empirical distribution defined by the latent codes of the training shapes, we introduce a quality checker of latent codes that defines a continuous regularized distribution of the latent space. This quality checker integrates a learned function that quantifies the geometric feasibility score of a synthetic shape and the spectral properties of a stiffness matrix that quantifies its physical stability score.

We show how to extend a state-of-the-art diffusion framework EDM [20] to integrate the data distribution and the quality checker for learning denoising networks. A key contribution is a principled approach that determines the trade-off parameters between loss terms of the data distribution and the quality checker at different noise levels.

We have evaluated the performance of GPLD3D on ShapeNet-v2 [6]. Experimental results show that GPLD3D significantly outperforms state-of-the-art shape generators on multiple metrics. We also present an ablation study to justify the importance of incorporating the quality checker and optimizing the hyperparameters of training losses.

2. Related Work

We discuss related work in four categories, that is, implicit shape generators, diffusion techniques, geometric feasibility evaluation, and physical stability assessment.

Implicit Shape Generators. Implicit shape representations [5, 12, 36, 48, 56] achieve state-of-the-art performance as they can generate objects with arbitrary topologies and infinite resolution. They originated from the MLP architecture used in DeepSDF [28], OccNet [29] and IM-Net [9]. Positional encoding [43] and periodic activation functions [37] can capture geometric details better than MLP applied directly to the coordinates.

However, a significant issue is that they cannot guarantee desired shape properties. For example, many synthetic shapes contain unrealistic geometric patterns that look different from training shapes. Moreover, some synthetic shapes are not even connected. Even connected shapes have thin hanging structures that are not physically stable. The goal of GPLD3D is to address these issues by developing a novel latent diffusion model.

Diffusion techniques. Diffusion models [39, 40] have outperformed GANs in image generation [14] and have many applications today. However, it is difficult to generate high-quality 3D shapes using diffusion models due to the fixed dimension constraint in the diffusion process. This fixed dimension issue is addressed under the latent diffusion approach [34] and the state-of-the-art shape generators [10, 12, 19, 36, 56, 59] mainly adopted this paradigm. They utilize a pre-trained implicit shape decoder and perform latent space diffusion. However, despite the improved visual appearances of synthetic 3D shapes both qualitatively and quantitatively, fundamental issues in geometric feasibility and physical stability still need to be solved. This motivates us to develop GPLD3D, which uses a scoring function that evaluates the geometric feasibility and physical stability of an arbitrary shape to regularize the diffusion procedure.

Along another line, EDM [20] presents seminal results on determining the hyperparameters of the diffusion model's training and sampling procedures. The key contributions of this paper are two bounds on the differences between Jacobians of logarithmic probabilities. The first considers the smoothed data distribution and the smoothed ground-truth distribution. The second considers the smoothed regularization distribution derived from the quality checker and the smoothed ground-truth distribution. These two bounds enable us to determine the tradeoff parameters among the loss terms at different noise levels.

GPLD3D is also relevant to classifier-guidance diffusion, which was introduced in [14] and recently applied to synthesizing human motions [21, 54]. However, these approaches require gradients between the classifier and the input latent codes, which are not applicable in our setting. In contrast, GPLD3D only requires a latent code score function. In addition, these approaches explore tradeoffs between diversity and quality during testing. On the contrary, GPLD3D improves the training phase of the denoising network by introducing regularization losses. These losses re-

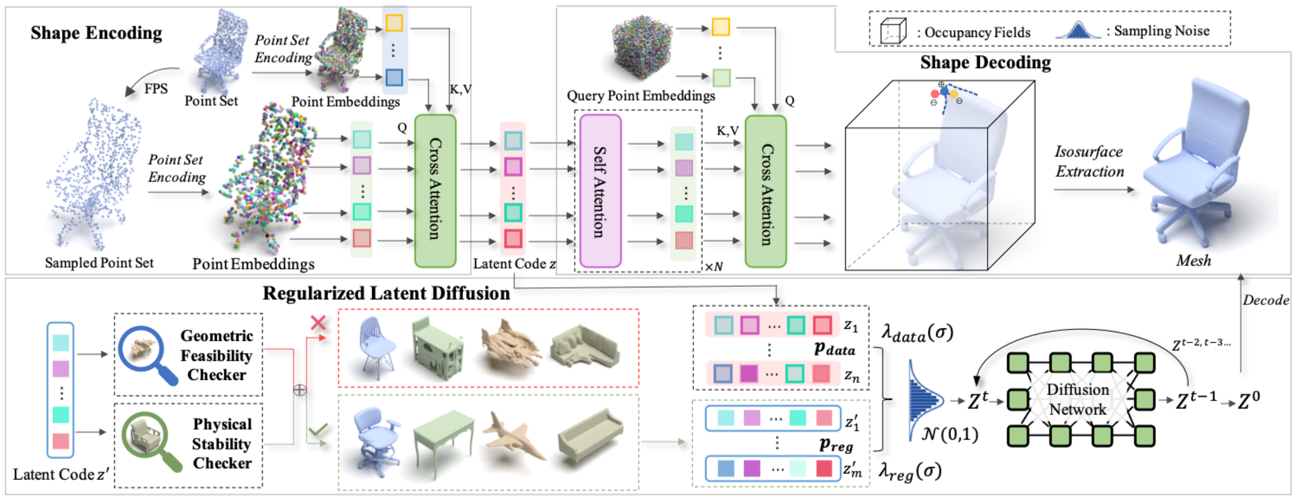


Figure 2. Overview of GPLD3D. It employs a quality checker that assesses the geometric feasibility and physical stability scores of synthetic shapes. This quality checker guides the diffusion procedure to sample in regions of the latent space that correspond to shapes that pass the quality checker and avoid regions that do not pass the quality checker. The backbone is 3DS2VS [56].

duce the variance of the learned denoising network to the log-gradient of the underlying smoothed ground-truth distribution.

Geometric Feasibility Evaluation. Visual data quality checking is a fundamental problem that has been studied extensively in the past [23, 26, 46, 47]. A standard approach is to perform local statistics analysis. However, such approaches are only applicable to data corrupted by a specific procedure. Recent approaches have taken the supervised learning paradigm [1, 2], but typically train a classifier or feature representation using class labels. Their performance in classifying high- and low-quality instances in each class is quite limited.

Another approach to distinguishing good and bad shapes is to employ discriminators used in GANs [15, 49]. However, such discriminators label all synthetic shapes as negative instances. This is different from our setting because we want to differentiate good and bad synthetic instances.

Our approach is conducted in a supervised learning manner. We find that learning to classify good and bad shapes based on class labels does not generalize well. On the contrary, our approach asks users to label good and bad synthetic shapes and learn a shape classifier to predict the quality of a synthetic shape. An interesting observation is that the classifier learned from synthetic shapes of one shape generator generalizes well to classifying synthetic shapes of other shape generators.

Physical Stability Assessment. Quantifying the physical stability of a 3D shape has been studied extensively in the computer graphics literature [41]. Many of these approaches [42, 60] focus on 3D printing applications. A common approach is to employ linear elasticity, which relates the displacement field and the external force field. [60] introduces a linear programming formulation that maximizes the objective function based on the stress field with respect to the external field in the worst case. However, performing this type of analysis is still time-consuming.

We find that the eigenvalues of the stiffness matrix are sufficient to determine the worst-case stress score of [60]. Therefore, we use MLPs that takes the eigenvalues of this matrix as input and outputs the predicted worst-case stress score. Experimental results show that this approach generalizes well to novel instances.

3. Problem Statement and Approach Overview

This section presents the problem statement and an overview of GPLD3D.

3.1. Problem Statement

The input to GPLD3D is a shape collection $\mathcal{S} = \{S_1, \cdots, S_n\} \subset \overline{\mathcal{S}}$ and a pre-trained decoder [56] $g^{\phi} : \mathcal{Z} := \mathbb{R}^d \rightarrow \overline{\mathcal{S}}$ that maps a latent code $\boldsymbol{z}$ to some 3D model $g^{\phi}(\boldsymbol{z})$. Denote by $\boldsymbol{z}_i$ the latent code of S_i in this pretrained decoder. Our goal is to train a diffusion model $\mathcal{D}^{\theta} : \mathcal{Z} \rightarrow \mathcal{Z}$ that satisfies two desired properties:

- The induced distribution $\mathcal{D}^{\theta}(\mathcal{N}_d)$ from the Gaussian distribution $\mathcal{N}_d$ aligns with the empirical distribution $\{\boldsymbol{z}_i\}$.
- When sampling $\boldsymbol{z} \sim \mathcal{D}^{\theta}(\mathcal{N}_d)$, $g^{\phi}(\boldsymbol{z})$ is physically stable and geometrically feasible.

3.2. Approach Overview

The key idea of GPLD3D is to develop a quality checker $q^{\psi}(g^{\phi}(\boldsymbol{z})) : \mathcal{Z} \rightarrow [0, 1]$ that evaluates the physical stability and geometric feasibility score of any synthetic shape $g^{\phi}(\boldsymbol{z})$. We then introduce a diffusion procedure that integrates the training instances and the quality checker. In the following, we highlight the main components of GPLD3D. Section 4 to Section 5 provide technical details.

Regularized latent diffusion. We introduce a regularized diffusion procedure that can integrate a quality checker in synthetic shapes. The key idea is to convert a quality checker into a regularization distribution. GPLD3D is based on a state-of-the-art diffusion framework [34], which reparameterizes the denoising network that matches the logarithmic probability gradients of the smoothed data distributions. Key contributions of GPLD3D are two bounds on differences between gradients of log-probabilities. The first considers the smoothed data distribution. The second considers the smoothed regularization distribution. These two bounds are used to determine the trade-off parameters of the loss terms that involve the data distribution and the regularization distribution at multiple noise levels. After training, the sampling procedure follows [56].

Quality check definition. GPLD3D employs a physical stability checker and a geometric feasibility checker. The physical stability checker is based on static analysis. Unlike performing static analysis for each new shape, we learn MLPs that takes the eigenvalues of a stiffness matrix and predicts its physical stability score. This network is trained on example shapes whose stability scores are derived from worst-case structural analysis. Regarding the geometric feasibility checker, we learn a classifier that predicts the quality of a synthetic shape. Training data are generated from labeled synthetic shapes of a shape generator.

4. Regularized Latent Diffusion

We begin by reviewing the standard diffusion framework in Section 4.1. We then present our regularized diffusion framework in Section 4.2.

4.1. Diffusion Review

The content of this section follows the state-of-the-art design space of diffusion models in EDM [20], and we refer to it for more details. Denote $p_{\text{data}} = \{\boldsymbol{z}_1, \cdots, \boldsymbol{z}_n\}$ as the empirical distribution defined by the latent codes of the training shapes. The aim of EDM is to learn a diffusion procedure that converts the Gaussian distribution of the latent space into the empirical distribution defined by $\overline{p}$. This is achieved by learning a denoising network $D_\theta(\boldsymbol{z};\sigma)$ where σ denotes the noise level. A good practice is to reparameterize D_θ in the following form:

$$D_\theta(\boldsymbol{z};\sigma) = c_{\text{skip}}(\sigma)\boldsymbol{z} + c_{\text{out}}(\sigma)F_\theta(c_{\text{in}}(\sigma)\boldsymbol{z}; c_{\text{noise}}(\sigma)) \quad (1)$$

where F_θ is a neural network to be trained, $c_{\text{skip}}(\sigma)$ modulates the skip-connection in the network, $c_{\text{in}}(\sigma)$ and $c_{\text{out}}(\sigma)$ scale the input and output magnitudes, and $c_{\text{noise}}(\sigma)$ maps noise level σ into a conditioning input for F_θ. Under this parameterization, the standard training loss $\mathbb{E}_{\sigma,\boldsymbol{x},\boldsymbol{n}}\lambda(\sigma)\|D_\theta(\boldsymbol{x}+\boldsymbol{n};\sigma)-\boldsymbol{x}\|^2$, where $\sigma \sim p_{\text{train}}$, $\boldsymbol{x} \sim p_{\text{data}}$, $\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0};\sigma^2 I)$, $\lambda(\sigma)$ is the weight of σ, corresponds to the following loss for F_θ:

$$\mathop{\mathbb{E}}_{\sigma,\boldsymbol{x},\boldsymbol{n}} \Big(\lambda(\sigma)c_{\text{out}}(\sigma)^2\|F_\theta\big(c_{\text{in}}(\sigma)(\boldsymbol{x}+\boldsymbol{n}); c_{\text{noise}}(\sigma)\big) - \frac{1}{c_{\text{out}}(\sigma)}\big(\boldsymbol{x} - c_{\text{skip}}(\sigma)(\boldsymbol{x}+\boldsymbol{n})\big)\|^2\Big) \quad (2)$$

EDM described a systematic way to set the hyperparameters $c_{\text{in}}(\sigma)$, $c_{\text{out}}(\sigma)$, $c_{\text{skip}}(\sigma)$, and $\lambda(\sigma)$. In addition, EDM [20] set $c_{\text{noise}}(\sigma) = \frac{1}{4}\log(\sigma)$.

After training F_θ, one can use the resulting $\mathcal{D}(\boldsymbol{x};\sigma)$ to progressively decode random noise into a latent code that follows the underlying distribution of p_{data}.

One issue with the above diffusion procedure is that it only uses p_{data} during training, which does not consider the geometric and physical properties of synthetic shapes that correspond to other latent codes. This motivates us to study regularized diffusion.

4.2. Regularized Diffusion

Regularized diffusion uses a quality checker $(q \circ g^\phi)(\boldsymbol{z})$ to improve the quality of $D_\theta(\boldsymbol{z};\sigma)$. The key idea is first to convert $(q \circ g^\phi)(\boldsymbol{z})$ into a continuous density function $p_{\text{reg}}(\boldsymbol{z})$ from which samples can be easily drawn. To do this, we simply set $p_{\text{reg}} = q \circ g^\phi$. In Section 5, we describe how to define $(q \circ g^\phi)$ and introduce a data structure that quickly evaluates $p_{\text{reg}}(\boldsymbol{z})$.

Given p_{reg}, we redefine the training loss on F_θ as

$$\min_\theta \mathop{\mathbb{E}}_{\sigma} \sum_{t\in\{\text{data},\text{reg}\}} f_t(\theta;\sigma) \quad (3)$$

$$f_t(\theta;\sigma) := \mathop{\mathbb{E}}_{\boldsymbol{n}} \mathop{\mathbb{E}}_{\boldsymbol{z}\sim p_t} \lambda_t(\sigma)\|D_\theta(\boldsymbol{z}+\boldsymbol{n};\sigma)-\boldsymbol{z}\|^2. \quad (4)$$

where $f_t(\theta)$ in (4) uses the reparameterization in (2). Besides $\lambda_{\text{data}}(\sigma)$ and $\lambda_{\text{reg}}(\sigma)$, $\lambda_{\text{in}}(\sigma)$, $\lambda_{\text{out}}(\theta)$, and $\lambda_{\text{skip}}(\theta)$, and $\lambda_{\text{noise}}(\theta)$ are shared between $f_{\text{data}}(\theta)$ and $f_{\text{reg}}(\theta)$.

Similarly to (2), the hyperparameters dictate the behavior of (3). The key difference between (3) and (2) is that we now have a continuous p_{reg} in addition to the discrete p_{data}. Therefore, a key task is to determine the relative ratio between $\lambda_{\text{data}}(\sigma)$ and $\lambda_{\text{reg}}(\sigma)$. To this end, let us begin by understanding the trade-offs between p_{reg} and p_{data} in light of the underlying continuous ground truth distribution p_{gt}. Introduce three smoothed distributions: $\forall t \in \{\text{data}, \text{reg}, gt\}$,

$$p_t(\boldsymbol{z};\sigma) := P(\boldsymbol{z}+\boldsymbol{n}|\boldsymbol{z}\sim p_t), \boldsymbol{n}\sim\mathcal{N}(\boldsymbol{0};\sigma^2 I).$$

Based on EDM [20] (Eqs(2,3)), minimizing each $f_t(\theta;\sigma)$ in isolation leads to

$$\nabla_{\boldsymbol{z}}\log p_t(\boldsymbol{z};\sigma) = \frac{D_\theta(\boldsymbol{z};\sigma)-\boldsymbol{z}}{\sigma^2}, \quad \forall t\in\{\text{data},\text{reg}\}. \quad (5)$$

Therefore, we can determine the relative ratio between $\lambda_{\text{data}}(\sigma)$ and $\lambda_{\text{reg}}(\sigma)$ based on the variances

$$\mathbb{V}_t(\sigma) = \mathop{\mathbb{E}}_{\boldsymbol{z}\sim p_t}\|\nabla_{\boldsymbol{z}}\log p_{gt}(\boldsymbol{z};\sigma) - \nabla_{\boldsymbol{z}}\log p_t(\boldsymbol{z};\sigma)\|^2.$$

It is easy to see that $\mathbb{V}_{\text{data}}(\sigma)$ is small for large σ. However, due to the discrete natural of p_{data}, $\mathbb{V}_{\text{data}}(\sigma)$ is large for small σ. The following theorem quantifies this intuition.

Theorem 1 *(**Informal**) Asymptotically,*

$$\mathbb{V}_{\text{data}}(\sigma) = O(\frac{1}{\sigma^4}). \tag{6}$$

Proof: See Appendix A.

The following theorem quantifies $\mathbb{V}_{\text{reg}}(\sigma)$ in the small regime of σ, where $\mathbb{V}_{\text{data}}(\sigma)$ is large.

Theorem 2 *(**Informal**) For small σ, we have*

$$\mathbb{V}_{\text{reg}}(\sigma) = \mathop{\mathbb{E}}_{\boldsymbol{z}\sim p_{\text{reg}}}\Big(O(\|\frac{\nabla_{\boldsymbol{z}} p_{\text{reg}}(\boldsymbol{z})}{p_{\text{reg}}(\boldsymbol{z})} - \frac{\nabla_{\boldsymbol{z}} p_{gt}(\boldsymbol{z})}{p_{gt}(\boldsymbol{z})}\|) + O(\sigma)\Big)^2 \tag{7}$$

where the constants in both $O(\cdot)$ only depend on $p_{gt}(\boldsymbol{z})$.

Proof: See Appendix B.

The difference between p_{reg} and p_{data} is that the range of p_{reg} is between 0 and 1. However, we observe that p_{reg}/p_{gt} is a smooth function. Therefore, the value of $\|\nabla_{\boldsymbol{z}} p_{\text{reg}}(\boldsymbol{z})/p_{\text{reg}}(\boldsymbol{z}) - \nabla_{\boldsymbol{z}} p_{gt}(\boldsymbol{z})/p_{gt}(\boldsymbol{z})\|$ is small. This means in the small σ regime, $\mathbb{V}_{\text{reg}}(\sigma) \ll \mathbb{V}_{\text{data}}(\sigma)$

Our goal is to balance the variances of the data term and the regularization term, that is, $\lambda_{\text{data}}(\sigma)\mathbb{V}_{\text{data}}(\sigma) = \lambda_{\text{reg}}(\sigma)\mathbb{V}_{\text{reg}}(\sigma)$. Applying (6) and (7), we set

$$\frac{\lambda_{\text{reg}}(\sigma)}{\lambda_{\text{data}}(\sigma)} = \frac{\mathbb{V}_{\text{data}}(\sigma)}{\mathbb{V}_{\text{reg}}(\sigma)} = \frac{c_{\text{ratio}}}{\sigma^4(c_{\text{off}} + \sigma)^2}. \tag{8}$$

Even (7) is only valid in the small σ regime, we find that (8) works for large σ as $\lambda_{\text{data}}(\sigma) \gg \lambda_{\text{reg}}(\sigma)$ in this regime.

We proceed to determine the absolute values of $\lambda_{\text{data}}(\sigma)$, $\lambda_{\text{reg}}(\sigma)$, $c_{\text{out}}(\sigma)$, $c_{\in}(\sigma)$, and $c_{\text{skip}}(\sigma)$ for different σ. To this end, we apply four modified principles of EDM [20]. They are 1) the variance of the weighted inputs for F_θ is 1, 2) the variance of the weighted fitting targets for F_θ is 1, 3) $c_{\text{skip}}(\sigma) = \arg\min_{c_{\text{skip}}(\sigma)} c_{\text{out}}(\sigma)$, and 4) balancing the denoising network at different σ. Due to space constraints, we defer the derivations to Appendix C and present the results below:

$$\lambda_{\text{data}}(\sigma) := \frac{\sigma_{\text{data}}^2 + \sigma^2 + \frac{c_{\text{ratio}}(\sigma_{\text{reg}}^2 + \sigma^2)}{\sigma^4(c_{\text{off}}+\sigma)^2}}{\Big(\sigma_{\text{data}}^2 + \frac{c_{\text{ratio}}\sigma_{\text{reg}}^2}{\sigma^4(c_{\text{off}}+\sigma)^2}\Big)\Big(\sigma^2 + \frac{c_{\text{ratio}}}{\sigma^2(c_{\text{off}}+\sigma)^2}\Big)},$$

$$\lambda_{\text{reg}}(\sigma) := \frac{c_{\text{ratio}}}{\sigma^4(c_{\text{off}} + \sigma)^2}\lambda_{\text{data}}(\sigma),$$

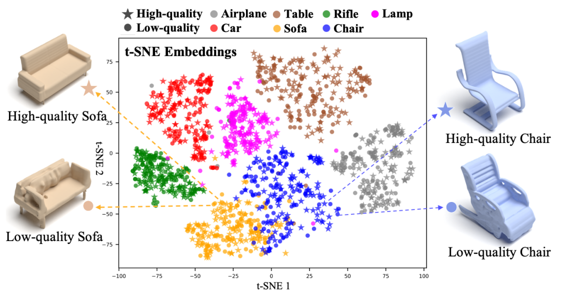


Figure 3. Visualization of t-SNE embeddings for features of 7 common categories. Classifiers trained to predict class labels cannot distinguish between high-quality and low-quality shapes.

$$c_{\text{in}}(\sigma) := \frac{1}{\sqrt{\sigma_{\text{data}}^2 + \sigma^2 + \frac{c_{\text{ratio}}(\sigma_{\text{reg}}^2+\sigma^2)}{\sigma^4(c_{\text{off}}+\sigma)^2}}}$$

$$c_{\text{out}}(\sigma) := \sqrt{\frac{\Big(\sigma_{\text{data}}^2 + \frac{c_{\text{ratio}}\sigma_{\text{reg}}^2}{\sigma^4(c_{\text{off}}+\sigma)^2}\Big)\Big(\sigma^2 + \frac{c_{\text{ratio}}}{\sigma^2(c_{\text{off}}+\sigma)^2}\Big)}{\sigma_{\text{data}}^2 + \sigma^2 + \frac{c_{\text{ratio}}(\sigma_{\text{reg}}^2+\sigma^2)}{\sigma^4(c_{\text{off}}+\sigma)^2}}}$$

$$c_{\text{skip}}(\sigma) := \frac{\sigma_{\text{data}}^2 + \frac{c_{\text{ratio}}\sigma_{\text{reg}}^2}{\sigma^4(c_{\text{off}}+\sigma)^2}}{\sigma_{\text{data}}^2 + \frac{c_{\text{ratio}}\sigma_{\text{reg}}^2}{\sigma^4(c_{\text{off}}+\sigma)^2} + \sigma^2 + \frac{c_{\text{ratio}}}{\sigma^2(c_{\text{off}}+\sigma)^2}}.$$

The same as EDM [20], we set $c_{\text{noise}}(\sigma) = \frac{1}{4}\log(\sigma)$. Note that when $c_{\text{ratio}} = 0$, the hyperparameters $\lambda_{\text{data}}(\sigma)$, $c_{\text{in}}(\sigma)$, $c_{\text{out}}(\sigma)$, and $c_{\text{skip}}(\sigma)$ above recover those from EDM.

5. Quality Check Function

This section introduces the quality check function $q^\psi(S), S \in \overline{\mathcal{S}}$. We define $q^\psi(S) = \alpha q_g^{\psi_g}(S) + (1-\alpha)q_p^{\psi_p}(S)$ as the combination of a geometric feasibility score $q_g^{\psi_g}(S)$ (Section 5.1) and a physical stability score $q_p^{\psi_p}(S)$ (Section 5.2). Section 5.3 describes how to compute $(q^\psi \circ g^\phi)(\boldsymbol{z})$ for regularized diffusion.

5.1. Geometric Feasibility Score

We develop a neural network $q_g^{\psi_g}(S)$ to predict the quality of a 3D shape $S \in \overline{\mathcal{S}}$. The network employs a variant of PointNet++ [30] and takes a point cloud representation (surfels that combine positions and normals) of S as input and outputs a quality score.

The fundamental challenge is how to train $q_g^{\psi_g}(S)$. An approach is to train $q_g^{\psi_g}$ to predict class labels, e.g., those of ShapeNetCore55. When evaluating synthetic shapes of a generator, the hope is that for a high-quality shape, the classifier will output a peaked distribution at the corresponding class. In contrast, for a low-quality shape, the predicted class distribution is more uniform. However, we find that this approach does not work in practice. For state-of-the-art shape generators [56], the pre-trained classifier always

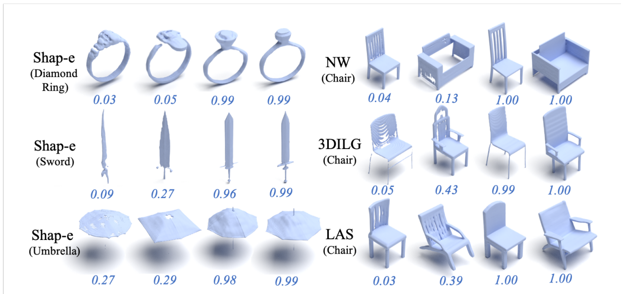


Figure 4. (Left) The trained shape quality predictor can generalize to novel classes generated by Shap-e [19]. (Right) The predictor generalizes well to synthetic shapes from other shape generators [18, 55, 59]. The predict scores are listed below each shape.

outputs peaked distributions for both high-quality and low-quality synthetic shapes (see Figure 3).

Therefore, we propose asking users to label synthetic shapes. To do this, we use synthetic shapes of [12, 28, 36, 56] in ShapeNetCore55 and ask users to annotate high- and low-quality instances. We train a shape classifier so that one branch predicts a label of each synthetic shape in 55 classes, and another branch predicts object quality, i.e., a high-quality class and a low-quality class for each object class. We find that the patterns that differentiate good and bad instances are small- to middle-scale features. Moreover, these patterns are shared between different generative models that share similar implicit representations. Therefore, we are able to learn a universal shape quality checker from synthetic shapes from a modest set of generative models and object classes. As shown in Figure 4, the learned shape quality predictor generalizes well to unseen object categories and synthetic shapes generated by other shape generators [18, 55, 59]. In both settings, the classifier is highly correlated with human evaluations (see Appendix D for more details).

5.2. Physical Stability Score

Our starting point is the worst-case structural analysis framework (WCSA) described in [60], which measures the stability of a 3D shape with respect to the stress field derived from the worst-case external force field. Specifically, given an implicit shape S, WCSA first converts it into a volumetric mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E})$. Using the linear elastic framework, WCSA expresses the displacement field as a linear mapping of the external force field. When the external force field is fixed, we can define a stress score as a function of the stress field, which is a linear function of the displacement field gradient. WCSA then solves a linear program to obtain the stress score with respect to the worst-case external force field. However, WCSA is too expensive to apply on large-scale shape collections.

In our experiments, we find that while the external force field in the worst case is difficult to predict, the stability score is strongly correlated with the eigenvalues of the stiffness matrix $K(S)$. For example, S is disconnected if and

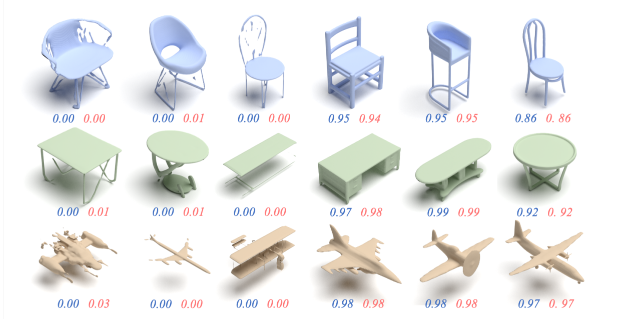


Figure 5. Visualization of synthetic shapes and their physical stability scores. The network generalizes to new shapes and object classes, where we list the simulated score (blue) and the predicted score (red) below each synthetic shape.

only if the seventh eigenvalue of $K(S)$ is zero. Therefore, we use a two-layer MLP that takes the eigenvalues of $K(S)$ as input to predict the stress score $s_{\text{sta}}(S)$ (the higher the more unstable). The network is trained from 6000 shapes with paired eigenvalues and WCSA results. We then define $q_p^{\psi_p}(S) = s_{\text{sta}}(S))/(\sigma + s_{\text{sta}}(S))$, where σ is taken as the quartile of the stress scores of the training shapes.

As shown in Figure 5, the trained network is well generalized to new shapes and object classes.

5.3. Scoring Function Approximation

When using the scoring function $q^{\psi}(S)$ defined in Section 5 to train the diffusion model, it requires decoding each latent $\boldsymbol{z}$ into a 3D shape for evaluation. To address this computational issue, we introduce an approach that approximates $(q^{\psi} \circ g^{\phi})(\boldsymbol{z})$ from samples of latent codes and their associated scores. As this is not the main contribution of this paper. We defer the technical details to the supp. material.

6. Experimental Results

We begin by introducing the experimental setup in Section 6.1. We then describe the experimental results and baseline comparisons in Section 6.2, Finally, Section 6.3 presents an ablation study.

6.1. Experimental Setup

Dataset. We perform an experimental evaluation on ShapeNet-v2 [6], which evaluated all baseline approaches. Figure 6 and Table 1 present qualitative and quantitative results, respectively. Similar to 3DILG [55] and 3DShape2VecSet [56], we report the results on 7 popular categories, i.e., table, car, chair, airplane, sofa, rifle, lamp. We also report the average results for all 55 categories.

Baseline approaches. We compare GPLD3D with four top performing baseline approaches, that is, 3DS2VS (3DShape2VecSet) [56], 3DILG [55], NW [18], and LAS [59]. In particular, 3DS2VS is our backbone, which corresponds to GPLD3D without enforcing quality checks.

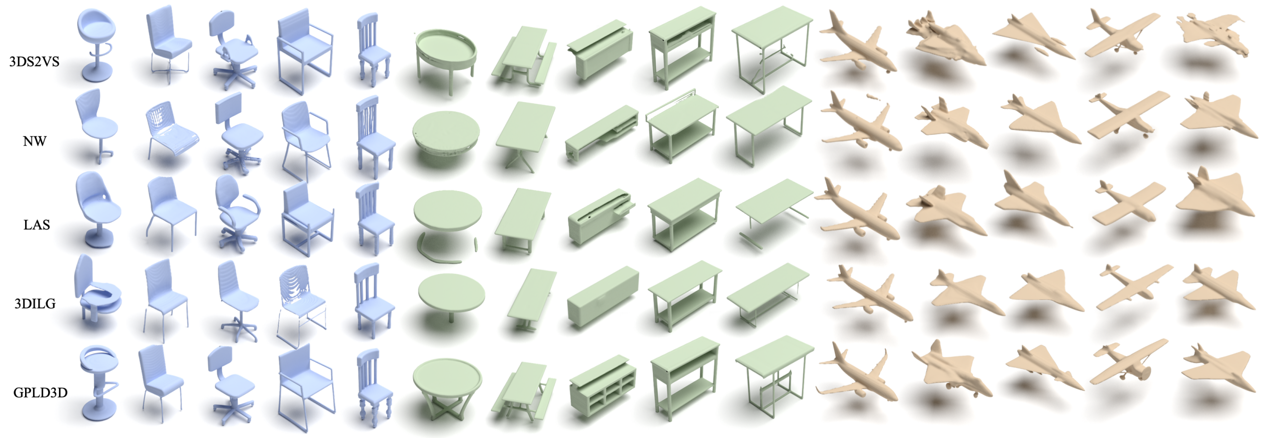


Figure 6. Category-conditional generation between GPLD3D and baselines. We find similar shapes of each method using OpenShape [25].

Metric	chair		table		airplane		sofa		rifle		lamp		car		all categories	
	FPD↓	KPD↓	FPD↓	KPD↓	FPD↓	KPD↓	FPD↓	KPD↓	FPD↓	KPD↓	FPD↓	KPD↓	FPD↓	KPD↓	FPD↓	KPD↓
3DS2VS [56]	0.63	0.48	0.49	0.35	0.35	0.38	0.58	0.41	0.67	0.68	0.87	0.51	0.52	0.32	0.67	0.60
LAS [59]	1.07	1.30	0.85	1.07	0.72	0.73	-	-	1.23	1.37	-	-	1.22	0.96	-	-
NW [18]	1.24	1.43	1.19	1.23	0.53	0.44	-	-	-	-	-	-	-	-	-	-
3DILG [55]	1.49	1.62	2.37	3.40	0.82	0.75	2.28	3.45	1.47	2.23	2.45	2.87	1.57	2.18	2.35	3.37
GPLD3D	**0.53**	**0.32**	**0.43**	**0.25**	**0.31**	**0.24**	**0.53**	**0.28**	**0.59**	**0.44**	**0.81**	**0.27**	**0.48**	**0.25**	**0.56**	**0.41**

Table 1. Quantitative comparisons between GPLD3D and baseline approaches under two metrics for evaluating shape generation results, i.e., FPD and KPD($\times 10^{-3}$). The sign '-' means the method does not release models trained on these categories.

The other baselines employ volumetric, octree, and implicit shape representations for shape generation.

Evaluation metrics. We employ four metrics for quantitative evaluations. The primary metrics are the Fréchet PointNet ++ Distance (FPD) and the Kernel PointNet ++ Distance (KPD) following [56]. Specifically, FPD and KPD is evaluated using the global feature embedding defined by the pre-trained shape category and quality classifier. In Appendix D, we show that this FPD score correlates with human evaluations much stronger than alternative FPD scores. The third metric, PSS, reports the mean physical stability score $q_p^{\psi_p}(S)$ among all synthetic shapes S. The fourth metric, PDS reports the percentage of synthetic shapes that are disconnected.

6.2. Analysis of Results

As shown in Figure 6, GPLD3D outperforms all baseline approaches in terms of visual appearance, geometric shapes, and structural feasibility. First, baseline approaches present disconnected synthetic shapes. This issue is significantly improved in GPLD3D. Moreover, baseline approaches present shapes that are not structurally meaningful which is also improved in GPLD3D. The synthetic shapes of GPLD3D are visually more appealing than the baseline approaches. All these results show the effectiveness of GPLD3D.

As shown in Table 1, GPLD3D outperforms baseline approaches in FPD / KPD scores. Specifically, the reductions from the highest performing baseline 3DS2VS are 16.5% / 31.7%. The reductions in the seven categories range from 7.7% / 21.9% (car) to 15.9% / 33.3% (chair), demonstrating the consistency of the improvements. The improvements in categories with complex geometrical and topological structures (chair and table) are greater than the improvements in categories with relatively simple topological structures (airplane, sofa and car). This behavior is expected because the quality checker is most effective on shapes with complex geometric, physical, and topological structures.

As shown in Table 2, GPLD3D outperforms the top baseline 3DS2VS by 13.1% and 24.5% in PSS and PDS across chair, table, airplane, and sofa. Among these, the improvements in PSS / PDS range from 9.9% / 13.1% to 23.9% / 36.6%. These numbers again show the effectiveness of the quality checker. Moreover, they indicate that interpolations induced by current generative models do not generalize well in structural shape properties.

As shown in Table 2, the PSS score of the training shapes is 0.829. This is because many input shapes have thin structures. In this sense, GPLD3D reduces the gap between the mean PSS score of 3DS2VS synthetic shapes from that of training shapes by 38.6%.

6.3. Ablation Study

No physical stability checking. We first study the effects of removing the physical stability checker. As shown in Table 2, the FPD score remains similar in all categories, showing that enforcing the geometric feasibility checker is sufficient to improve the geometric feasibility of the gener-

Metric	chair			table			airplane			sofa			mean		
	FPD↓	PSS↑	PDS↑	FPD↓	PSS↑	PDS↑	FPD↓	PSS↑	PDS↑	FPD↓	PSS↑	PDS↑	FPD↓	PSS↑	PDS↑
3DS2VS [56]	0.63	0.681	0.818	0.49	0.718	0.840	0.35	0.727	0.835	0.58	0.837	0.918	0.51	0.741	0.853
w/o Phy	0.56	0.699	0.832	0.44	0.729	0.851	**0.31**	0.738	0.845	0.55	0.852	0.922	0.46	0.755	0.862
w/o Geo	0.58	0.708	0.856	0.47	0.737	0.862	0.32	0.750	0.870	0.57	0.863	0.940	0.48	0.765	0.882
w/o HP-Opt	0.55	0.711	0.844	0.45	0.734	0.853	0.33	0.751	0.852	0.54	0.859	0.936	0.47	0.763	0.871
Full Model	**0.53**	**0.723**	**0.859**	**0.43**	**0.746**	**0.870**	**0.31**	**0.758**	**0.879**	**0.53**	**0.876**	**0.948**	**0.45**	**0.775**	**0.889**
Training	-	0.812	-	-	0.814	-	-	0.791	-	-	0.901	-	-	0.829	-

Table 2. Comparing our full model with 3DS2VS [56] and various ablated cases. Evaluation metrics include FPD, PSS and PSD.

ated shapes. However, adding a physical stability checker does not hurt geometric feasibility.

The two physical stability scores, PSS and PDS, drop in this case, that is, by 8.9% and 24.3%, respectively. This shows that simply checking geometric feasibility, which is based on the analysis of patterns of geometric features, is insufficient to enforce physical stability that involves global structural properties.

Compared to 3DS2VS, which does not enforce any quality check, enforcing the geometric feasibility check improves the two metrics that are relevant to physical stability, i.e., by 5.4% and 6.1% in PSS and PDS, respectively. This is expected, as many geometrically infeasible shapes are also physically unstable. Examples include shapes with disconnected components and thin and unrealistic structures.

No geometric feasibility checker. Next, we study the effects of dropping the geometric feasibility checker. In this case, the FPD score drops considerably (that is, by 6.7% on average) across four categories, showing that simply enforcing the physical stability checker is insufficient to improve the geometric feasibility of generated shapes. This is expected because the physical stability checker, which assesses global structural properties, does not consider local geometric features.

The two physical stability scores, PSS and PDS, drop slightly, i.e., by 4.3% and 5.9%, respectively. This shows that adding the geometric feasibility checker does not compete with improving the physical stability. We can understand this behavior because the percentage of physically stable but geometrically infeasible shapes is relatively small.

Compared to 3DS2VS, which does not enforce quality checks, the physical stability checker improves FPD by 5.9%. An explanation is that many physically unstable shapes are also geometrically infeasible. Examples include disconnected shapes and shapes with thin structures.

No hyper-parameter optimization. We study the effects of other ways of setting $\lambda_{\text{data}}(\sigma)$ and $\lambda_{\text{reg}}(\sigma)$. One way is to set $\lambda_{\text{data}} = \frac{1}{\sigma^2}$ as in prior work [27, 38, 40] and let $\lambda_{\text{reg}}(\sigma) = \frac{c_{\text{ratio}}}{\sigma^2}$, so that the regularization distribution is treated the same as the data distribution. Although in this case all metrics are still better than without the involvement of the quality checker, there are salient gaps from our approach of setting hyperparameters. Specfically, hyper-parameter optimization accounts for 30% of improvements in FPD/KPD and PSS from without using the qualify checker. These results show the importance of our approach in derive hyper-parameters from the analysis in Theorem 1 and Theorem 2.

7. Limitations

One limitation of GPLD3D is that it cannot guarantee the geometric feasibility and physical stability of synthetic shapes. One reason is that, when learning the diffusion model, the training procedure only sees a few million latent codes. Although this number is significantly more than the number of training shapes, it is still sparse in the high-dimensional latent space. One way to address this issue is to study how to develop local approximations of the quality checker and integrate information provided by such local approximations into the diffusion procedure.

Another limitation is that the quality of the decoder still limits its performance. It requires that the reconstruction errors of the testing shapes be small. An interesting question is how to improve the quality of the decoder by installing geometric and physical priors. This allows us to reduce the latent dimension, improve the latent distribution of training shapes, and consequently enhances the efficiency and generalization of training the diffusion model.

8. Conclusions and Future Work

This paper has introduced GPLD3D, a novel latent diffusion model for the generation of 3D shapes. GPLD3D incorporates a quality checker that assesses the geometric feasibility and physical stability of a synthetic shape in the diffusion procedure. We introduce a principled approach to determine the tradeoff parameters between the loss terms of the training data and the quality checker at multiple noise levels. The experimental results show that GPLD3D outperforms the baseline approaches in ShapeNet-v2.

Our work opens many avenues for future research. On the quality checker side, it is interesting to incorporate other priors, e.g., topological, part, and semantic priors. When defining physical stability, we can also explore using physical engines to perform physical simulations and evaluate the affordance via humans. We also want to extend GPLD3D for the generation of 3D scenes. On the theoretical side, it is exciting to quantify the behavior of (6) and (7) at extreme values of σ. These results will help to determine better trade-off parameters for very small and very large σ. Finally, while an advantage of GPLD3D is that it does not use quality checker gradients, it would be interesting to see how to integrate the gradients into the diffusion procedure.

References

[1] Ilyass Abouelaziz, Mohammed El Hassouni, and Hocine Cherifi. A convolutional neural network framework for blind mesh visual quality assessment. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 755–759. IEEE, 2017. 3

[2] Ilyass Abouelaziz, Aladine Chetouani, Mohammed El Hassouni, Longin Jan Latecki, and Hocine Cherifi. No-reference mesh visual quality assessment via ensemble of convolutional neural networks and compact multi-linear pooling. *Pattern Recognition*, 100:107174, 2020. 3

[3] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning representations and generative models for 3d point clouds. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 40–49. PMLR, 2018. 2

[4] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 214–223. PMLR, 2017. 1

[5] Alexandre Boulch and Renaud Marlet. Poco: Point convolution for surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6302–6314, 2022. 2

[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, 2015. 2, 6, 15

[7] Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. *arXiv preprint arXiv:2303.11396*, 2023. 17, 18

[8] Kevin Chen, Christopher B Choy, Manolis Savva, Angel X Chang, Thomas Funkhouser, and Silvio Savarese. Text2shape: Generating shapes from natural language by learning joint embeddings. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*, pages 100–116. Springer, 2019. 15

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

[10] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander G. Schwing, and Liangyan Gui. Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 4456–4465. IEEE, 2023. 2

[11] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander G. Schwing, and Liang-Yan Gui. Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4456–4465, 2023. 15, 17

[12] Gene Chou, Yuval Bahat, and Felix Heide. Diffusion-sdf: Conditional generative modeling of signed distance functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2262–2272, 2023. 2, 6

[13] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*, pages 628–644. Springer, 2016. 15

[14] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 8780–8794, 2021. 2

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

[16] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, 2020. 1

[17] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 15, 16

[18] Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3d shape generation. In *SIGGRAPH Asia 2022 Conference Papers, SA 2022, Daegu, Republic of Korea, December 6-9, 2022*, pages 24:1–24:9. ACM, 2022. 6, 7, 14

[19] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. 2, 6, 14

[20] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022. 2, 4, 5, 14

[21] Korrawe Karunratanakul, Konpat Preechakul, Supasorn Suwajanakorn, and Siyu Tang. Guided motion diffusion for controllable human motion synthesis. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 2151–2162. IEEE, 2023. 2

[22] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. 1

[23] Guillaume Lavoué. A multiscale metric for 3d mesh visual quality assessment. In *Computer graphics forum*, pages 1427–1437. Wiley Online Library, 2011. 3

[24] Or Litany, Alexander M. Bronstein, Michael M. Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages

1886–1895. Computer Vision Foundation / IEEE Computer Society, 2018. 2

[25] Minghua Liu, Ruoxi Shi, Kaiming Kuang, Yinhao Zhu, Xuanlin Li, Shizhong Han, Hong Cai, Fatih Porikli, and Hao Su. Openshape: Scaling up 3d shape representation towards open-world understanding, 2023. 7

[26] Anish Mittal, Rajiv Soundararajan, and Alan C. Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal Process. Lett.*, 20(3):209–212, 2013. 3

[27] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 8162–8171. PMLR, 2021. 8

[28] Jeong Joon Park, Peter R. Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 165–174. Computer Vision Foundation / IEEE, 2019. 2, 6

[29] Songyou Peng, Michael Niemeyer, Lars M. Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part III*, pages 523–540. Springer, 2020. 2

[30] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5099–5108, 2017. 5

[31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 15

[32] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J. Black. Generating 3D faces using convolutional mesh autoencoders. In *European Conference on Computer Vision (ECCV)*, pages 725–741, 2018. 2

[33] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1530–1538. JMLR.org, 2015. 1

[34] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 10674–10685. IEEE, 2022. 2, 4

[35] Philip Sedgwick. Pearson's correlation coefficient. *Bmj*, 345, 2012. 14

[36] J. Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20875–20886, 2023. 2, 6

[37] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Red Hook, NY, USA, 2020. Curran Associates Inc. 2

[38] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 8

[39] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11895–11907, 2019. 2

[40] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 2, 8

[41] Tommaso Sorgente, Silvia Biasotti, Gianmarco Manzini, and Michela Spagnuolo. A survey of indicators for mesh quality assessment. In *Computer Graphics Forum*, pages 461–483. Wiley Online Library, 2023. 3

[42] Ondrej Stava, Juraj Vanek, Bedrich Benes, Nathan Carr, and Radomír Měch. Stress relief: Improving structural strength of 3d printable objects. *ACM Trans. Graph.*, 31(4), 2012. 3

[43] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Red Hook, NY, USA, 2020. Curran Associates Inc. 2

[44] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6306–6315, 2017. 1

[45] Jordan Voas, Yili Wang, Qixing Huang, and Raymond Mooney. What is the best automated metric for text to motion generation? *arXiv preprint arXiv:2309.10248*, 2023. 14

[46] Kai Wang, Fakhri Torkhani, and Annick Montanvert. A fast roughness-based approach to the assessment of 3d mesh visual quality. *Computers & Graphics*, 36(7):808–818, 2012. 3

[47] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4): 600–612, 2004. 3

[48] Francis Williams, Zan Gojcic, Sameh Khamis, Denis Zorin, Joan Bruna, Sanja Fidler, and Or Litany. Neural fields as learnable kernels for 3d reconstruction. In *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 18500–18510, 2022. 2

[49] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in neural information processing systems*, 29, 2016. 3

[50] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, page 82–90, Red Hook, NY, USA, 2016. Curran Associates Inc. 2

[51] Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Liang Pan Jiawei Ren, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, Dahua Lin, and Ziwei Liu. Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 17

[52] Xingguang Yan, Liqiang Lin, Niloy J Mitra, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Shapeformer: Transformer-based shape completion via sparse representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6239–6249, 2022. 1

[53] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge J. Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 4540–4549. IEEE, 2019. 1, 2

[54] Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. Physdiff: Physics-guided human motion diffusion model. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 15964–15975. IEEE, 2023. 2

[55] Biao Zhang, Matthias Nießner, and Peter Wonka. 3dilg: Irregular latent grids for 3d generative modeling. In *NeurIPS*, 2022. 1, 6, 7, 14, 15

[56] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Trans. Graph.*, 42(4), 2023. 1, 2, 3, 4, 5, 6, 7, 8, 14, 15, 16, 17

[57] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. 17

[58] Xin-Yang Zheng, Yang Liu, Peng-Shuai Wang, and Xin Tong. Sdf-stylegan: Implicit sdf-based stylegan for 3d shape generation. In *Comput. Graph. Forum (SGP)*, 2022. 1

[59] Xin-Yang Zheng, Hao Pan, Peng-Shuai Wang, Xin Tong, Yang Liu, and Heung-Yeung Shum. Locally attentional sdf diffusion for controllable 3d shape generation. *ACM Transactions on Graphics (SIGGRAPH)*, 42(4), 2023. 2, 6, 7, 14

[60] Qingnan Zhou, Julian Panetta, and Denis Zorin. Worst-case structural analysis. *ACM Trans. Graph.*, 32(4), 2013. 3, 6

[61] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2242–2251. IEEE Computer Society, 2017. 1

GPLD3D: Latent Diffusion of 3D Shape Generative Models by Enforcing Geometric and Physical Priors

Supplementary Material

A. Proof of Theorem 1

As any probability distribution can be approximated by a mixture of Gaussian with a prescribed approximation error, it is sufficient to prove Theorem 1 in the setting where p_{gt} is given a mixture of Gaussians.

Theorem 3 ***(Formal statement of Theorem 1)*** *Consider a mixture of Gaussians with K mixture components, where each component $(w_i, \boldsymbol{m}_i, \Sigma_i)$ is given by its mixture weight w_i, the mean $\boldsymbol{m}_i$, and the covariance matrix Σ_i:*

$$p_{gt}(\boldsymbol{z}) := \sum_{i=1}^{K} \frac{w_i}{(2\pi)^{\frac{d}{2}}|\Sigma_i|^{\frac{1}{2}}} \exp\big(-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{m}_i)^T\Sigma^{-1}(\boldsymbol{z}-\boldsymbol{m}_i)\big). \tag{9}$$

Let p_{data} be the empirical distribution defined by n random samples from $p_{gt}(\boldsymbol{z})$. Then

$$\mathop{\mathbb{E}}_{p_{\text{data}}} \mathop{\mathbb{E}}_{\boldsymbol{z}\sim p_{gt}} \|\nabla_{\boldsymbol{z}} p_{\text{data}}(\boldsymbol{z},\sigma) - \nabla_{\boldsymbol{z}} p_{gt}(\boldsymbol{z},\sigma)\|^2 = O(\frac{1}{\sigma^4}). \tag{10}$$

Proof of Theorem 3. Note that

$$p_{gt}(\boldsymbol{z},\sigma) := \sum_{i=1}^{K} \frac{w_i \exp\big(-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{m}_i)^T\Sigma_i(\sigma)^{-1}(\boldsymbol{z}-\boldsymbol{m}_i)\big)}{(2\pi)^{\frac{d}{2}}|\Sigma_i(\sigma)|^{\frac{1}{2}}},$$

where

$$\Sigma_i(\sigma) := \Sigma_i + \sigma^2 \cdot I_d$$

Denote $p_{\text{data}} = \{\boldsymbol{z}_1, \cdots, \boldsymbol{z}_n\}$. Then

$$p_{\text{data}}(\boldsymbol{z},\sigma) := \frac{1}{n}\sum_{i=1}^{n} \frac{1}{(2\pi)^{\frac{d}{2}}\sigma^d}\exp(-\frac{\|\boldsymbol{z}\|^2}{2\sigma^2}).$$

It follows that

$$\nabla_{\boldsymbol{z}} \log(p_{gt}(\boldsymbol{z},\sigma)) = -\sum_{i=1}^{K} \frac{w_i\Sigma_i(\sigma)^{-1}(\boldsymbol{z}-\boldsymbol{m}_i)}{(2\pi)^{\frac{d}{2}} p_{gt}(\boldsymbol{z},\sigma)|\Sigma_i(\sigma)|^{\frac{1}{2}}} \tag{11}$$

$$\nabla_{\boldsymbol{z}} \log(p_{\text{data}}(\boldsymbol{z},\sigma)) = -\frac{\sum\limits_{i=1}^{n}\exp\big(-\frac{\|\boldsymbol{z}-\boldsymbol{z}_i\|^2}{2\sigma^2}\big)(\boldsymbol{z}-\boldsymbol{z}_i)}{\sigma^2\sum\limits_{i=1}^{n}\exp\big(-\frac{\|\boldsymbol{z}-\boldsymbol{z}_i\|^2}{2\sigma^2}\big)} \tag{12}$$

To estimate the order of (10) in σ, we consider the regime of σ where $\sigma \gg \max\limits_{1\le i\le K}(\|\boldsymbol{m}_i\|, \|\Sigma_i\|)$. In this regime, we have when $\boldsymbol{z} \sim p_{gt}$,

$$p_{gt}(\boldsymbol{z},\sigma) \approx \sum_{i=1}^{K} \frac{w_i\big(1-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{m}_i)^T\Sigma_i(\sigma)^{-1}(\boldsymbol{z}-\boldsymbol{m}_i)\big)}{(2\pi)^{\frac{d}{2}}|\Sigma_i(\sigma)|^{\frac{1}{2}}} \tag{13}$$

$$\exp(-\frac{\|\boldsymbol{z}-\boldsymbol{z}_i\|^2}{2\sigma^2}) \approx 1 - \frac{\|\boldsymbol{z}-\boldsymbol{z}_i\|^2}{2\sigma^2} \tag{14}$$

Substituting (13) into (11), we have

$$\begin{aligned}&\nabla_{\boldsymbol{z}} \log(p_{gt}(\boldsymbol{z},\sigma))\\ \approx& -\frac{\sum\limits_{i=1}^{K}\frac{w_i}{|\Sigma_i(\sigma)|^{\frac{1}{2}}} + \frac{1}{2}\sum\limits_{i=1}^{K}\frac{w_i(\boldsymbol{z}-\boldsymbol{m}_i)^T\Sigma_i(\sigma)^{-1}(\boldsymbol{z}-\boldsymbol{m}_i)}{|\Sigma_i(\sigma)|^{\frac{1}{2}}}}{\Big(\sum\limits_{i=1}^{K}\frac{w_i}{|\Sigma_i(\sigma)|^{\frac{1}{2}}}\Big)^2}\\ &\cdot\sum_{i=1}^{K}\frac{w_i\Sigma_i(\sigma)^{-1}(\boldsymbol{z}-\boldsymbol{m}_i)}{|\Sigma_i(\sigma)|^{\frac{1}{2}}}\end{aligned} \tag{15}$$

In the large σ regime, we have

$$\|\Sigma_i(\sigma)^{-2}\| = O(\frac{1}{\sigma^2}).$$

Therefore,

$$\nabla_{\boldsymbol{z}} \log(p_{gt}(\boldsymbol{z},\sigma)) \approx -\frac{\sum\limits_{i=1}^{K}\frac{w_i\Sigma_i(\sigma)^{-1}(\boldsymbol{z}-\boldsymbol{m}_i)}{|\Sigma_i(\sigma)|^{\frac{1}{2}}}}{\sum\limits_{i=1}^{K}\frac{w_i}{|\Sigma_i(\sigma)|^{\frac{1}{2}}}} \tag{16}$$

Similarly, substituting (14) into (12), we have

$$\begin{aligned}&\nabla_{\boldsymbol{z}} \log(p_{\text{data}}(\boldsymbol{z},\sigma))\\ \approx& -\frac{1}{\sigma^2 n^2}\sum_{i,j=1}^{n}(1-\frac{\|\boldsymbol{z}-\boldsymbol{z}_i\|^2}{2\sigma^2})(1+\frac{\|\boldsymbol{z}-\boldsymbol{z}_j\|^2}{2\sigma^2})(\boldsymbol{z}-\boldsymbol{z}_i)\\ \approx& -\frac{1}{\sigma^2 n}\sum_{i=1}^{n}(1-\frac{\|\boldsymbol{z}-\boldsymbol{z}_i\|^2}{2\sigma^2}+\frac{\|\boldsymbol{z}-\boldsymbol{z}_i\|^2}{2\sigma^2})(\boldsymbol{z}-\boldsymbol{z}_i)\\ =& -\frac{\boldsymbol{z}}{\sigma^2}+\frac{1}{\sigma^2 n}\sum_{i=1}^{n}\boldsymbol{z}_i\end{aligned} \tag{17}$$

Combing (16) and (17), we have

$$\|\nabla_{\boldsymbol{z}} \log(p_{gt}(\boldsymbol{z},\sigma)) - \nabla_{\boldsymbol{z}} \log(p_{\text{data}}(\boldsymbol{z},\sigma))\| = O(\frac{1}{\sigma^4}) \tag{18}$$

□

B. Proof of Theorem 2

We then quantify the difference between $\nabla_{\boldsymbol{z}} \log(p_{\text{reg}}(\boldsymbol{z};\sigma))$ and $\nabla_{\boldsymbol{z}} \log(p_{gt}(\boldsymbol{z};\sigma))$. Suppose we have the following.

Note that

$$\begin{aligned}&\nabla_{\boldsymbol{z}} \log(p_{\text{reg}}(\boldsymbol{z};\sigma))\\ =&\frac{1}{p_{\text{reg}}(\boldsymbol{z};\sigma)}\nabla_{\boldsymbol{z}} p_{\text{reg}}(\boldsymbol{z};\sigma)\\ =&\frac{\int_{\boldsymbol{y}} \nabla_{\boldsymbol{z}} p_{\text{reg}}(\boldsymbol{z}-\boldsymbol{y})\exp\big(-\frac{\|\boldsymbol{y}\|^2}{2\sigma^2}\big)d\boldsymbol{y}}{\int_{\boldsymbol{y}} p_{\text{reg}}(\boldsymbol{z}-\boldsymbol{y})\exp\big(-\frac{\|\boldsymbol{y}\|^2}{2\sigma^2}\big)d\boldsymbol{y}} \qquad (19)\end{aligned}$$

Introduce

$$\boldsymbol{e}_1(\boldsymbol{z},\boldsymbol{y}) = \frac{\nabla_{\boldsymbol{z}} p_{\text{reg}}(\boldsymbol{z}-\boldsymbol{y})}{p_{\text{reg}}(\boldsymbol{z})} - \frac{\nabla_{\boldsymbol{z}} p_{gt}(\boldsymbol{z}-\boldsymbol{y})}{p_{gt}(\boldsymbol{z})}, \qquad (20)$$

$$e_2(\boldsymbol{z},\boldsymbol{y}) = \frac{p_{\text{reg}}(\boldsymbol{z}-\boldsymbol{y})}{p_{\text{reg}}(\boldsymbol{z})} - \frac{p_{gt}(\boldsymbol{z}-\boldsymbol{y})}{p_{gt}(\boldsymbol{z})}. \qquad (21)$$

Applying Taylor series, we obtain

$$\begin{aligned}\boldsymbol{e}_1(\boldsymbol{z},\boldsymbol{y}) = \sum_{k=0}^{+\infty}\frac{(-1)^k}{k!}\Big(&\frac{1}{p_{\text{reg}}(\boldsymbol{z})}\langle\frac{\partial^{k+1}p_{\text{reg}}(\boldsymbol{z})}{\partial^{k+1}\boldsymbol{z}},\otimes^k\boldsymbol{y}\rangle\\ &-\frac{1}{p_{gt}(\boldsymbol{z})}\langle\frac{\partial^{k+1}p_{gt}(\boldsymbol{z})}{\partial^{k}\boldsymbol{z}},\otimes^{k+1}\boldsymbol{y}\rangle\Big),\end{aligned}$$

and

$$\begin{aligned}e_2(\boldsymbol{z},\boldsymbol{y}) = \sum_{k=1}^{+\infty}\frac{(-1)^k}{k!}\Big(&\frac{1}{p_{\text{reg}}(\boldsymbol{z})}\langle\frac{\partial^{k}p_{\text{reg}}(\boldsymbol{z})}{\partial^{k}\boldsymbol{z}},\otimes^k\boldsymbol{y}\rangle\\ &-\frac{1}{p_{gt}(\boldsymbol{z})}\langle\frac{\partial^{k}p_{gt}(\boldsymbol{z})}{\partial^{k}\boldsymbol{z}},\otimes^{k}\boldsymbol{y}\rangle\Big).\end{aligned}$$

Define

$$c_k(\boldsymbol{z}) := \Big\|\frac{\frac{\partial^k p_{\text{reg}}(\boldsymbol{z})}{\partial^k \boldsymbol{z}}}{p_{\text{reg}}(\boldsymbol{z})} - \frac{\frac{\partial^k p_{gt}(\boldsymbol{z})}{\partial^k \boldsymbol{z}}}{p_{gt}(\boldsymbol{z})}\Big\|.$$

It follows that

$$\|\boldsymbol{e}_1(\boldsymbol{z},\boldsymbol{y})\| \le \sum_{k=0}^{+\infty}\frac{c_{k+1}(\boldsymbol{z})\|\boldsymbol{y}\|^k}{k!}, \qquad (22)$$

and

$$|e_2(\boldsymbol{z},\boldsymbol{y})| \le \sum_{k=1}^{+\infty}\frac{c_{k}(\boldsymbol{z})\|\boldsymbol{y}\|^k}{k!}, \qquad (23)$$

Rearranging (20) and (21), we arrive at

$$\begin{aligned}&\nabla_{\boldsymbol{z}} p_{\text{reg}}(\boldsymbol{z}-\boldsymbol{y})\\ =&\boldsymbol{e}_1(\boldsymbol{z},\boldsymbol{y})p_{\text{reg}}(\boldsymbol{z}) + \frac{p_{\text{reg}}(\boldsymbol{z})}{p_{gt}(\boldsymbol{z})}\nabla_{\boldsymbol{z}} p_{gt}(\boldsymbol{z}-\boldsymbol{y}), \qquad (24)\end{aligned}$$

and

$$\begin{aligned}&p_{\text{reg}}(\boldsymbol{z}-\boldsymbol{y})\\ =&\boldsymbol{e}_2(\boldsymbol{z},\boldsymbol{y})p_{\text{reg}}(\boldsymbol{z}) + \frac{p_{\text{reg}}(\boldsymbol{z})}{p_{gt}(\boldsymbol{z})}p_{gt}(\boldsymbol{z}-\boldsymbol{y}), \qquad (25)\end{aligned}$$

Substituting (25) and (24) into (19), we obtain

$$\begin{aligned}&\nabla_{\boldsymbol{z}} \log(p_{\text{reg}}(\boldsymbol{z};\sigma))\\ =&\frac{\int_{\boldsymbol{y}}\big(\nabla_{\boldsymbol{z}} p_{gt}(\boldsymbol{z}-\boldsymbol{y}) + p_{gt}(\boldsymbol{z})\boldsymbol{e}_1(\boldsymbol{z},\boldsymbol{y})\big)\exp(-\frac{\|\boldsymbol{y}\|^2}{2\sigma^2})d\boldsymbol{y}}{\int_{\boldsymbol{y}}\big(p_{gt}(\boldsymbol{z}-\boldsymbol{y}) + p_{gt}(\boldsymbol{z})e_2(\boldsymbol{z},\boldsymbol{y})\big)\exp(-\frac{\|\boldsymbol{y}\|^2}{2\sigma^2})d\boldsymbol{y}}\end{aligned} \qquad (26)$$

Introduce

$$\delta_k = \frac{1}{\sigma^{d+k}}\int_{\boldsymbol{y}\in\mathbb{R}^d}\|\boldsymbol{y}\|^k\exp\big(-\frac{\|\boldsymbol{y}\|^2}{2\sigma^2}\big)d\boldsymbol{y}.$$

It is easy to check that

$$\delta_k = \int_{\boldsymbol{y}\in\mathbb{R}^d}\|\boldsymbol{y}\|^k\exp\big(-\frac{\|\boldsymbol{y}\|^2}{2}\big)d\boldsymbol{y}$$

where Γ is the Chi-distribution. Let

$$\boldsymbol{c}_g(\boldsymbol{z},\sigma) := \frac{1}{\sigma^d}\int_{\boldsymbol{y}}\nabla_{\boldsymbol{z}} p_{gt}(\boldsymbol{z}-\boldsymbol{y})\exp(-\frac{\|\boldsymbol{y}\|^2}{2\sigma^2})d\boldsymbol{y} \qquad (27)$$

$$c(\boldsymbol{z},\sigma) := \frac{1}{\sigma^d}\int_{\boldsymbol{y}} p_{gt}(\boldsymbol{z}-\boldsymbol{y})\exp(-\frac{\|\boldsymbol{y}\|^2}{2\sigma^2})d\boldsymbol{y} \qquad (28)$$

Combing (22), (23), (27), (28), and (26), we have that in the small regime σ,

$$\begin{aligned}&\|\nabla_{\boldsymbol{z}} \log(p_{\text{reg}}(\boldsymbol{z};\sigma)) - \nabla_{\boldsymbol{z}} \log(p_{gt}(\boldsymbol{z};\sigma))\|\\ \le&\Big\|\frac{\int_{\boldsymbol{y}}\big(\nabla_{\boldsymbol{z}} p_{gt}(\boldsymbol{z}-\boldsymbol{y}) + p_{gt}(\boldsymbol{z})\boldsymbol{e}_1(\boldsymbol{z},\boldsymbol{y})\big)\exp(-\frac{\|\boldsymbol{y}\|^2}{2\sigma^2})d\boldsymbol{y}}{\int_{\boldsymbol{y}}\big(p_{gt}(\boldsymbol{z}-\boldsymbol{y}) + p_{gt}(\boldsymbol{z})e_2(\boldsymbol{z},\boldsymbol{y})\big)\exp(-\frac{\|\boldsymbol{y}\|^2}{2\sigma^2})d\boldsymbol{y}}\\ &-\frac{\int_{\boldsymbol{y}}\nabla_{\boldsymbol{z}} p_{gt}(\boldsymbol{z}-\boldsymbol{y})\exp(-\frac{\|\boldsymbol{y}\|^2}{2\sigma^2})d\boldsymbol{y}}{\int_{\boldsymbol{y}} p_{gt}(\boldsymbol{z}-\boldsymbol{y})\exp(-\frac{\|\boldsymbol{y}\|^2}{2\sigma^2})d\boldsymbol{y}}\Big\|\\ \le&p_{gt}(\boldsymbol{z})\frac{c(\boldsymbol{z},\sigma)c_1(\boldsymbol{z})\delta_0 + \sigma\big(\epsilon_2 c(\boldsymbol{z},\sigma) + \epsilon_1\|\boldsymbol{c}_g(\boldsymbol{z},\sigma)\|\big)}{c(\boldsymbol{z},\sigma)\big(c(\boldsymbol{z},\sigma) - p_{gt}(\boldsymbol{z})\sigma\epsilon_1\big)}\end{aligned} \qquad (29)$$

where

$$\epsilon_1 := \sum_{k=1}^{\infty}\frac{c_k(\boldsymbol{z})}{k!}\sigma^{k-1}\delta_k$$

$$\epsilon_2 := \sum_{k=1}^{\infty}\frac{c_{k+1}(\boldsymbol{z})}{k!}\sigma^{k-1}\delta_k$$

□

C. Details on the Optimal Hyper-Parameters

We adopt three principles from [56] to determine $\lambda_{\text{in}}(\sigma)$, $\lambda_{\text{out}}(\sigma)$, and $\lambda_{\text{skip}}(\sigma)$. First, we want the variance of the weighted inputs to be 1, i.e.,

$$\begin{aligned}&\operatorname*{Var}_{\boldsymbol{z}\sim p_{\text{data}},\boldsymbol{n}}\big(c_{\text{in}}(\sigma)(\boldsymbol{z}+\boldsymbol{n})\big)\\&+\frac{\lambda_{\text{reg}}(\sigma)}{\lambda_{\text{data}}(\sigma)}\operatorname*{Var}_{\boldsymbol{z}\sim p_{\text{reg}},\boldsymbol{n}}\big(c_{\text{in}}(\sigma)(\boldsymbol{y}+\boldsymbol{n})\big)=1 \qquad (30)\end{aligned}$$

Note that

$$\operatorname*{Var}_{\boldsymbol{z}\sim p_{\text{data}},\boldsymbol{n}}(c_{\text{in}}(\sigma)(\boldsymbol{z}+\boldsymbol{n})=c_{\text{in}}(\sigma)^2(\sigma_{\text{data}}^2+\sigma^2) \qquad (31)$$

$$\operatorname*{Var}_{\boldsymbol{z}\sim p_{\text{reg}},\boldsymbol{n}}(c_{\text{in}}(\sigma)(\boldsymbol{y}+\boldsymbol{n})=c_{\text{in}}(\sigma)^2(\sigma_{\text{reg}}^2+\sigma^2) \qquad (32)$$

Substituting (31) and (32) into (30), we obtain

$$c_{\text{in}}(\sigma)=\frac{1}{\sqrt{\sigma_{\text{data}}^2+\sigma^2+\frac{c_{\text{ratio}}(c_{\text{reg}}^2+\sigma^2)}{\sigma^2(c_{\text{off}}+\sigma)^2}}} \qquad (33)$$

Second, we constrain that the weighted fitting targets to have unit variance, i.e.,

$$\begin{aligned}&\frac{1}{c_{\text{out}}(\sigma)^2}\Big(\operatorname*{Var}_{\boldsymbol{z}\sim p_{\text{data}},\boldsymbol{n}}\big(\boldsymbol{z}-c_{\text{skip}}(\sigma)(\boldsymbol{z}+\boldsymbol{n})\big)\\&+\frac{\lambda_{\text{reg}}(\sigma)}{\lambda_{\text{data}}(\sigma)}\operatorname*{Var}_{\boldsymbol{z}\sim p_{\text{reg}},\boldsymbol{n}}\big(\boldsymbol{z}-c_{\text{skip}}(\sigma)(\boldsymbol{z}+\boldsymbol{n})\big)\Big)=1 \qquad (34)\end{aligned}$$

Note that

$$\begin{aligned}&\operatorname*{Var}_{\boldsymbol{z}\sim p_{\text{data}},\boldsymbol{n}}\big((\boldsymbol{z}-c_{\text{skip}}(\sigma)(\boldsymbol{z}+\boldsymbol{n})\big)\\&=(1-c_{\text{skip}}(\sigma))^2\sigma_{\text{data}}^2+c_{\text{skip}}(\sigma)^2\sigma^2 \qquad (35)\\&\operatorname*{Var}_{\boldsymbol{z}\sim p_{\text{reg}},\boldsymbol{n}}\big(\boldsymbol{z}-c_{\text{skip}}(\sigma)(\boldsymbol{z}+\boldsymbol{n})\big)\\&=(1-c_{\text{skip}}(\sigma))^2\sigma_{\text{reg}}^2+c_{\text{skip}}(\sigma)^2\sigma^2 \qquad (36)\end{aligned}$$

Substituting (35) and (36) into (34), we obtain

$$\begin{aligned}c_{\text{out}}^2(\sigma)=&(1-c_{\text{skip}}(\sigma))^2(\sigma_{\text{data}}^2+\frac{c_{\text{ratio}}\sigma_{\text{reg}}^2}{\sigma^2(c_{\text{off}}+\sigma)^2})\\&+c_{\text{skip}}(\sigma)^2(\sigma^2+\frac{c_{\text{ratio}}}{(c_{\text{off}}+\sigma)^2}) \qquad (37)\end{aligned}$$

Third, we select $c_{\text{skip}}(\sigma)$ to minimize $c_{\text{out}}(\sigma)$, i.e.,

$$c_{\text{skip}}(\sigma)=\operatorname*{argmin}_{c_{\text{skip}}(\sigma)}\ c_{\text{out}}(\sigma)^2.$$

This leads to

$$c_{\text{skip}}(\sigma)=\frac{\sigma_{\text{data}}^2+\frac{c_{\text{ratio}}\sigma_{\text{reg}}^2}{\sigma^2(c_{\text{off}}+\sigma)^2}}{\sigma_{\text{data}}^2+\frac{c_{\text{ratio}}\sigma_{\text{reg}}^2}{\sigma^2(c_{\text{off}}+\sigma)^2}+\sigma^2+\frac{c_{\text{ratio}}}{(c_{\text{off}}+\sigma)^2}} \qquad (38)$$

Substituting (38) into (37), we arrive at

$$c_{\text{out}}(\sigma)=\sqrt{\frac{\big(\sigma_{\text{data}}^2+\frac{c_{\text{ratio}}\sigma_{\text{reg}}^2}{\sigma^2(c_{\text{off}}+\sigma)^2}\big)\big(\sigma^2+\frac{c_{\text{ratio}}}{(c_{\text{off}}+\sigma)^2}\big)}{\sigma_{\text{data}}^2+\sigma^2+\frac{c_{\text{ratio}}(\sigma_{\text{reg}}^2+\sigma^2)}{\sigma^2(c_{\text{off}}+\sigma)^2}}} \qquad (39)$$

Now we determine $\lambda_{\text{data}}(\sigma)$ and $\lambda_{\text{reg}}(\sigma)$. Note that their relative values are determined as

$$\frac{\lambda_{\text{reg}}(\sigma)}{\lambda_{\text{data}}(\sigma)}=\frac{c_{\text{ratio}}}{\sigma^4(c_{\text{off}}+\sigma)^2}.$$

Therefore, it remains to determine the relative values of $\lambda_{\text{data}}(\sigma)$ for different σ. To this end, we follow EDM (pp 29) [20] and enforce that

$$\lambda_{\text{data}}(\sigma)c_{\text{out}}(\sigma)^2=1.$$

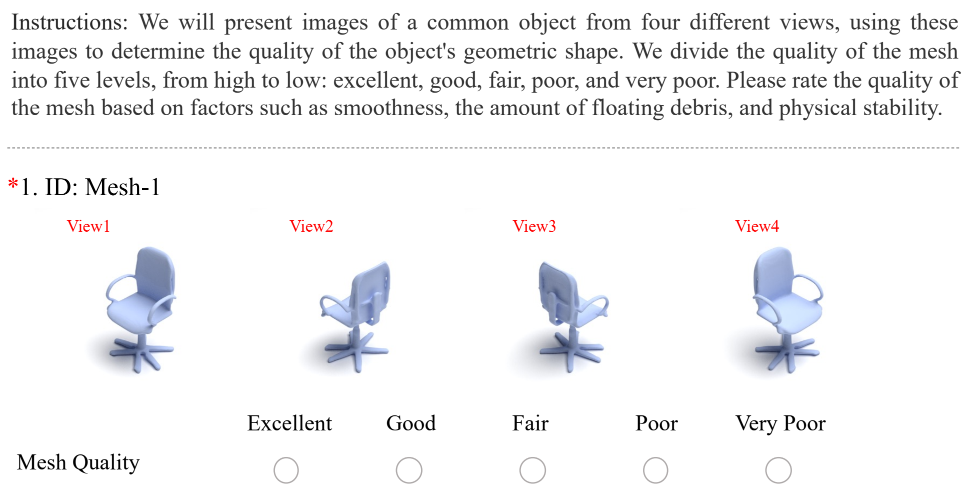
This leads to

$$\lambda_{\text{data}}(\sigma)=\frac{\sigma_{\text{data}}^2+\sigma^2+\frac{c_{\text{ratio}}(\sigma_{\text{reg}}^2+\sigma^2)}{\sigma^4(c_{\text{off}}+\sigma)^2}}{\big(\sigma_{\text{data}}^2+\frac{c_{\text{ratio}}\sigma_{\text{reg}}^2}{\sigma^4(c_{\text{off}}+\sigma)^2}\big)\big(\sigma^2+\frac{c_{\text{ratio}}}{\sigma^2(c_{\text{off}}+\sigma)^2}\big)}.$$

D. Correlations between the Quality Checker with Human Evaluations

Following [45], we utilize Pearson's Correlation Coefficient [35] to assess the correlation between the mesh quality score and the FPD score with human evaluations, measuring the linear relationship between metrics. We use other shape generators (i.e., NW [18], 3DILG [55], and LAS [59]) to generate shapes for the categories of chair, airplane, and table. Additionally, we employ shap-e [19] to generate objects in five unseen categories (umbrella, diamond ring, sword, shoes, and clothing). For each generator, we generate 200 synthetic objects for the known categories and 100 synthetic objects for the unseen categories, each using different random seeds. We asked 20 ordinary users to score the mesh quality of the generated shapes. An example of the user interface for human evaluation is shown in Figure 7. We evaluated the correlation of people with Quality Score and Fréchet PointNet++ Distance (FPD). FPD* indicates that PointNet++ used in this metric is trained with ShapeNetCore55 as well as additional labeled synthetic shapes. The 'Quality Score' is obtained from the quality prediction branch of the classifier.

We evaluated FPD only at the model level, as it requires distributional statistics over multiple samples, preventing sample-level calculations. As shown in Table 3, Quality Score achieved 0.708 Pearson's in all samples, indicating a strong correlation with human evaluations. FPD* and FPD achieve 0.702 Pearson's and 0.625 Pearson's respectively, indicating FPD* correlates with human evaluations much stronger than alternative FPD. Note that Shap-e is not included in calculating FPD because there is no ground-truth data distribution.


Instructions: We will present images of a common object from four different views, using these images to determine the quality of the object's geometric shape. We divide the quality of the mesh into five levels, from high to low: excellent, good, fair, poor, and very poor. Please rate the quality of the mesh based on factors such as smoothness, the amount of floating debris, and physical stability.

*1. ID: Mesh-1

	Excellent	Good	Fair	Poor	Very Poor
Mesh Quality	○	○	○	○	○

Figure 7. Rating selection UI. Instructions for raters in human evaluations are located above the rating selection.

	Quality Score	FPD	FPD*
Model Level	-	-0.625	-0.702
Sample Level	0.708	-	-

Table 3. Person correlations calculated with human judgments for Quality Score and FPD. Sample-level correlations are calculated on every individual samples, while model-level correlations are determined using all samples generated by a specific model.

E. Implementation Details

E.1. Quality Checker Computation Acceleration

To address this computational issue, we train neural networks to directly predict geometric feasibility and physical stability scores to avoid decoding a latent code during training time. We used a transformer to predict the score, the architecture of which is similar to that of the shape decoder. As shown in Figure 8, the transformer consists of a series of self-attention blocks. This approach avoids decoding the latent code into a shape and then sample a point cloud for classification. We trained different networks to predict the geometric feasibility and physical stability scores.

E.2. Training Details

For category-conditioned generation, we use the ShapeNet-v2 dataset [6] as a benchmark and utilize the training/val splits in [55, 56]. And each shape is converted to a watertight mesh and then normalized to fit within its bounding box ($[-1,1]^3$). We use 3DS2VS [56] as our backbone. For shape encoding, the point cloud of size 2048 is input to the pre-trained encoder to get the latent code $\boldsymbol{z}t \in p_{\text{data}}$. For regularized diffusion, we use the 3DS2VS latent diffusion model [56] as our generator to obtain the latent code $\boldsymbol{z}^{'}$, which is then fed into the quality checker to obtain a quality score $q^{\psi}(g^{\phi}(\boldsymbol{z}))$. In this work, we pass the latent code $\boldsymbol{z}^{'} \in p_{\text{reg}}$ when the quality score q^{ψ} exceeds 0.9. We initialize our diffusion model with weights [56] and train it on 8 V100 using AdamW optimizer with batch size of 192 for 6,400 epochs. In each batch, we sample $2/3$ latent codes from p_{data} and $1/3$ latent codes from p_{reg}. The learning rate is set to $4e-5$ and gradually decreased using the cosine decay schedule. For category-conditioned generation, objects from all categories are used for training, and the category label is mapped to an embedding. This embedding is then input into the denoising neural network as conditional information. We employ the default configurations for the EDM hyperparameters and set additional hyperparameters $c_{\text{ratio}} = 10^{-2}$ and $c_{\text{off}} = 1.0$ in this paper. During sampling, we obtain the final latent code via 40 denoising steps.

F. Additional Results

F.1. Image-conditioned Generation

For the single view object reconstruction, we use the 2D rendering dataset provided by 3D-R2N2 [13]. In this dataset, each shape is rendered in RGB images with dimensions of 224 × 224 pixels, captured from 24 randomly selected angles. We also adopt CLIP [31] as the image encoder and compare GPLD3D with 3DS2VS [56] and AtlasNet [17] in Figure 9.

F.2. Text-conditioned Generation

For text-driven shape generation, we use the Text2shape dataset [8], which provides descriptions of the chair and table categories in ShapeNet. We adopt CLIP [31] as the text encoder and compare GPLD3D with 3DS2VS [56] and SDFusion [11]. For GPLD3D and 3DS2VS, we generate the

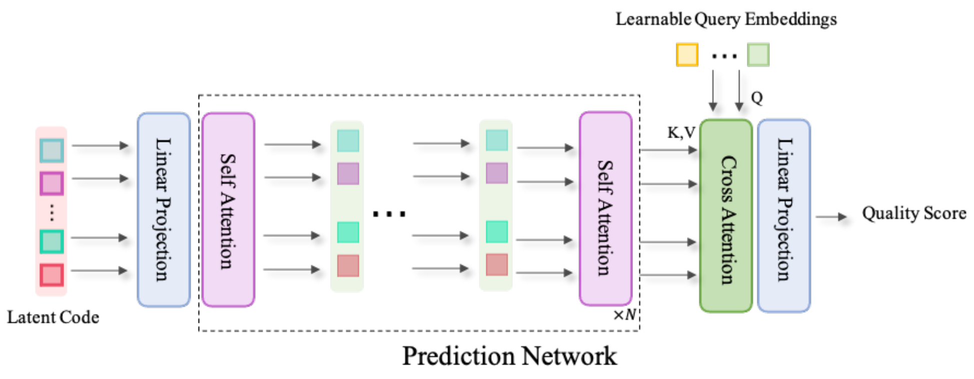


Figure 8. The architecture of the score prediction network.

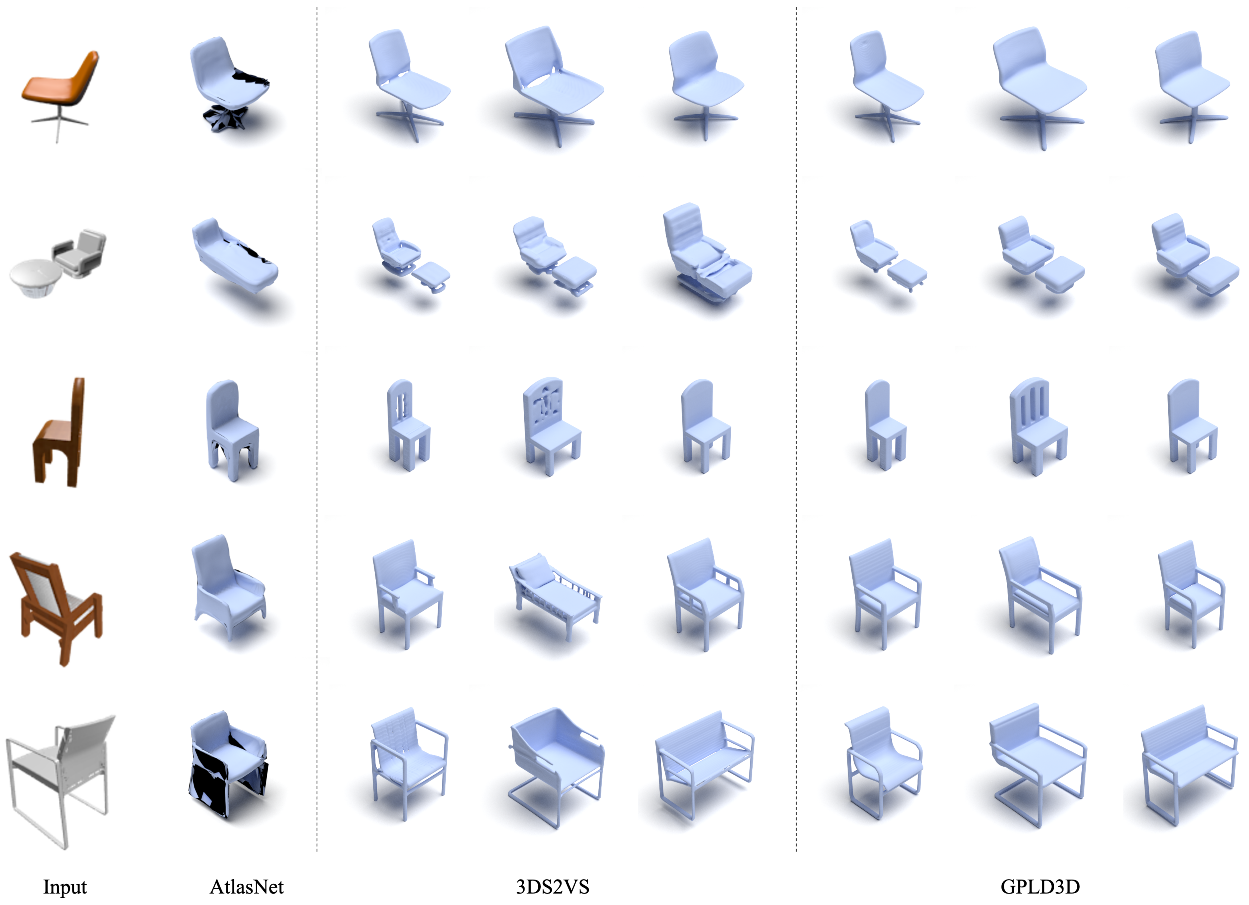


Figure 9. Image-conditioned 3D shape generation. We compare GPLD3D with AtlasNet [17] and 3DS2VS [56] on the 3D-R2N2 (chair) dataset. GPLD3D generates high-quality shapes with more details.

meshes by running Marching Cubes on a 128^3 grid. The results of our text-conditioned generation model can be found in Figure 10. Qualitatively, GPLD3D generates shapes of better quality and with fewer geometric artifacts.

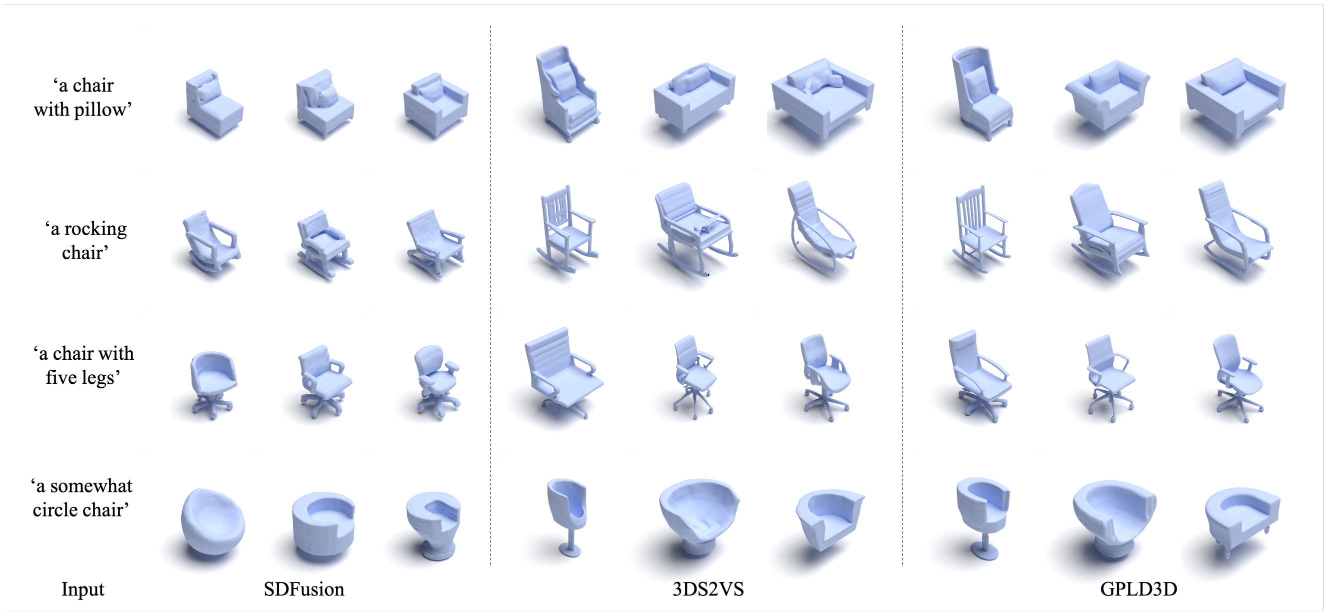


Figure 10. Text-conditioned 3D shape generation. We compare GPLD3D with SDFusion [11] and 3DS2VS [56] on the Text2Shape dataset for chairs. GPLD3D generates shapes with more details and fewer geometric issues, while complying with the description.

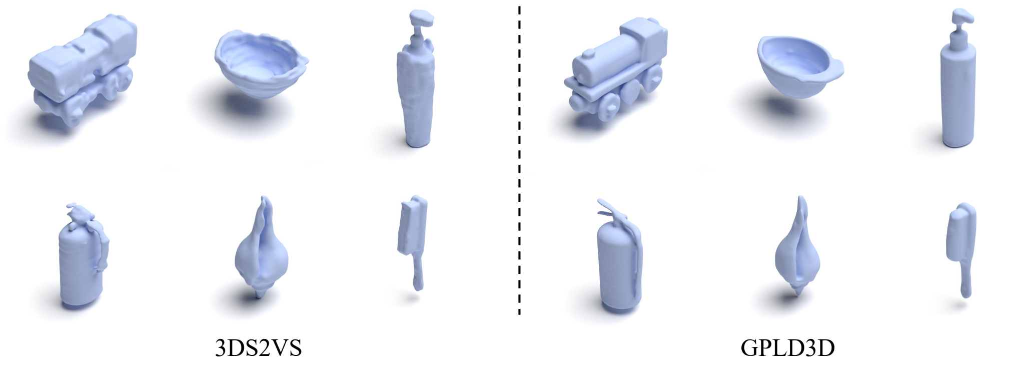


Figure 11. Unconditional generation between GPLD3D and 3DS2VS on OmniObject3D [51] dataset.

F.3. Additional experiment on the OmniObject3D

We conducted an additional experiment on the OmniObject3D [51] dataset, which comprises 6,000 scanned objects in 190 categories. We train an unconditional model using this dataset and Fig. 11 shows some qualitative results compared with 3DS2VS [56].

F.4. 3D Shape Texturing

We showcase an application that uses our text-conditioned GPLD3D F.2 to generate 3D shapes with detailed geometry, and then employs a depth-conditioned text-to-image 2D diffusion model [7, 57] to generate consistent textures. The results are shown in Figure 12, We can generate high-quality object meshes based on the input text prompts and obtain textured objects through the texture generation algorithm.

F.5. More Visual Galleries on Different Categories

We provided more visual results for category-conditioned generation across different categories (Chair in Figure 13, Table in Figure 14, Airplane in Figure 15, Sofa in Figure 17, Car in Figure 16, Lamp in Figure 18).

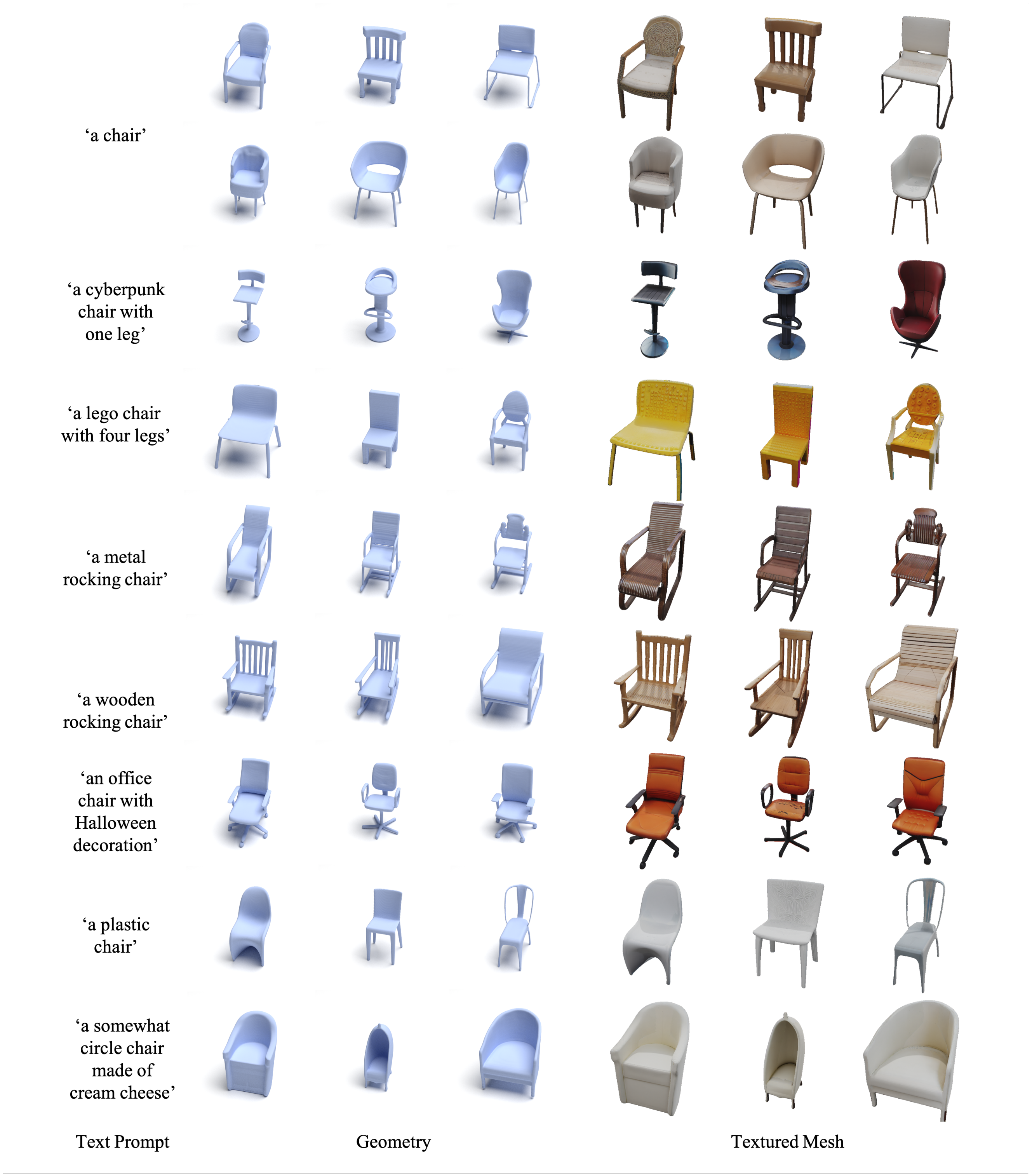

'a chair'
'a cyberpunk chair with one leg'
'a lego chair with four legs'
'a metal rocking chair'
'a wooden rocking chair'
'an office chair with Halloween decoration'
'a plastic chair'
'a somewhat circle chair made of cream cheese'
Text Prompt
Geometry
Textured Mesh

Figure 12. Text-conditioned 3D shape texturing. Firstly, we generate an object mesh using text-conditioned GPLD3D, and then texture the synthetic shapes with Text2Tex [7]. Given a text prompt, the pipeline can generate high-quality and diverse textured meshes.

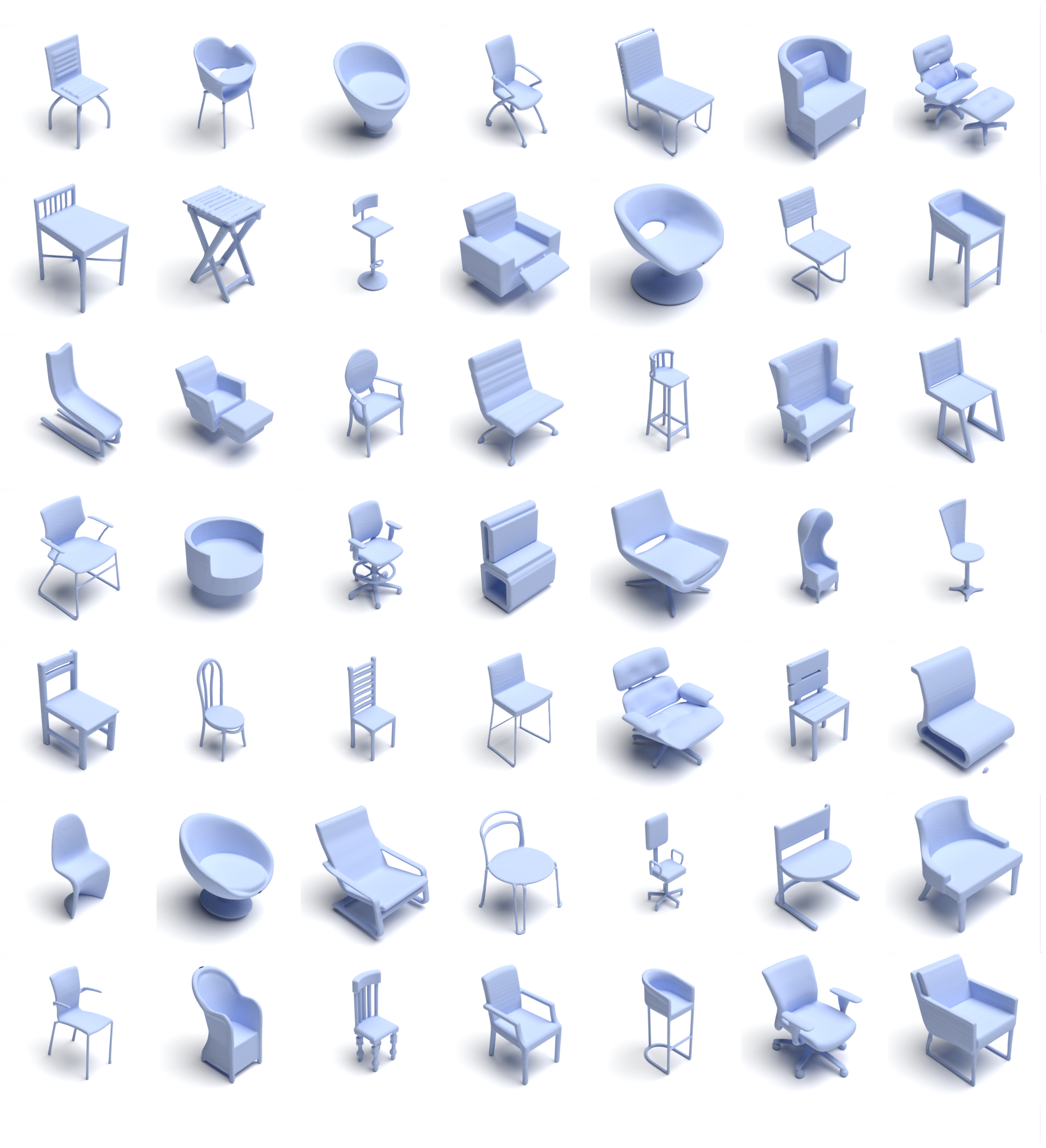


Figure 13. Gallery of chairs generated by our method.

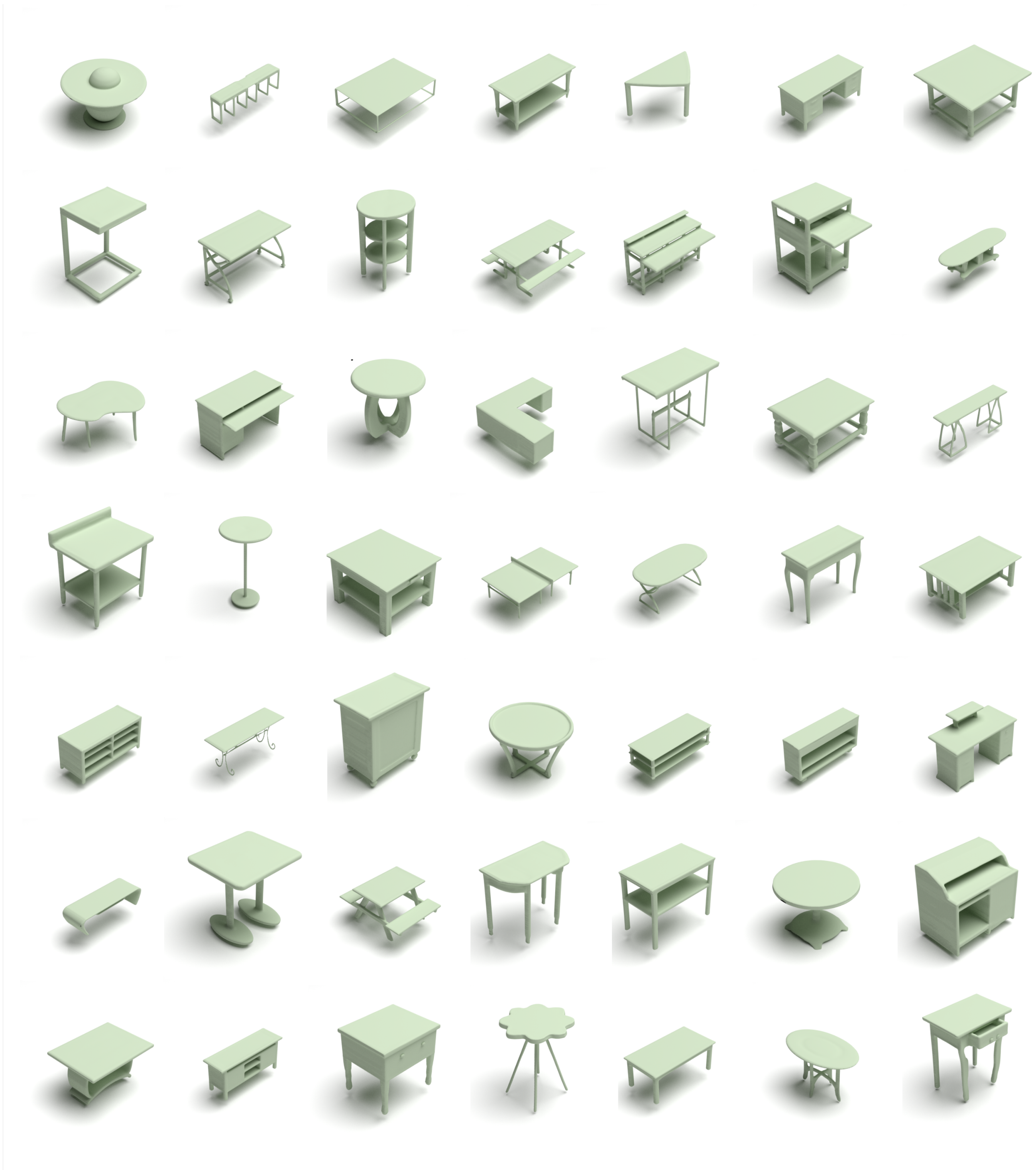


Figure 14. Gallery of tables generated by GPLD3D.

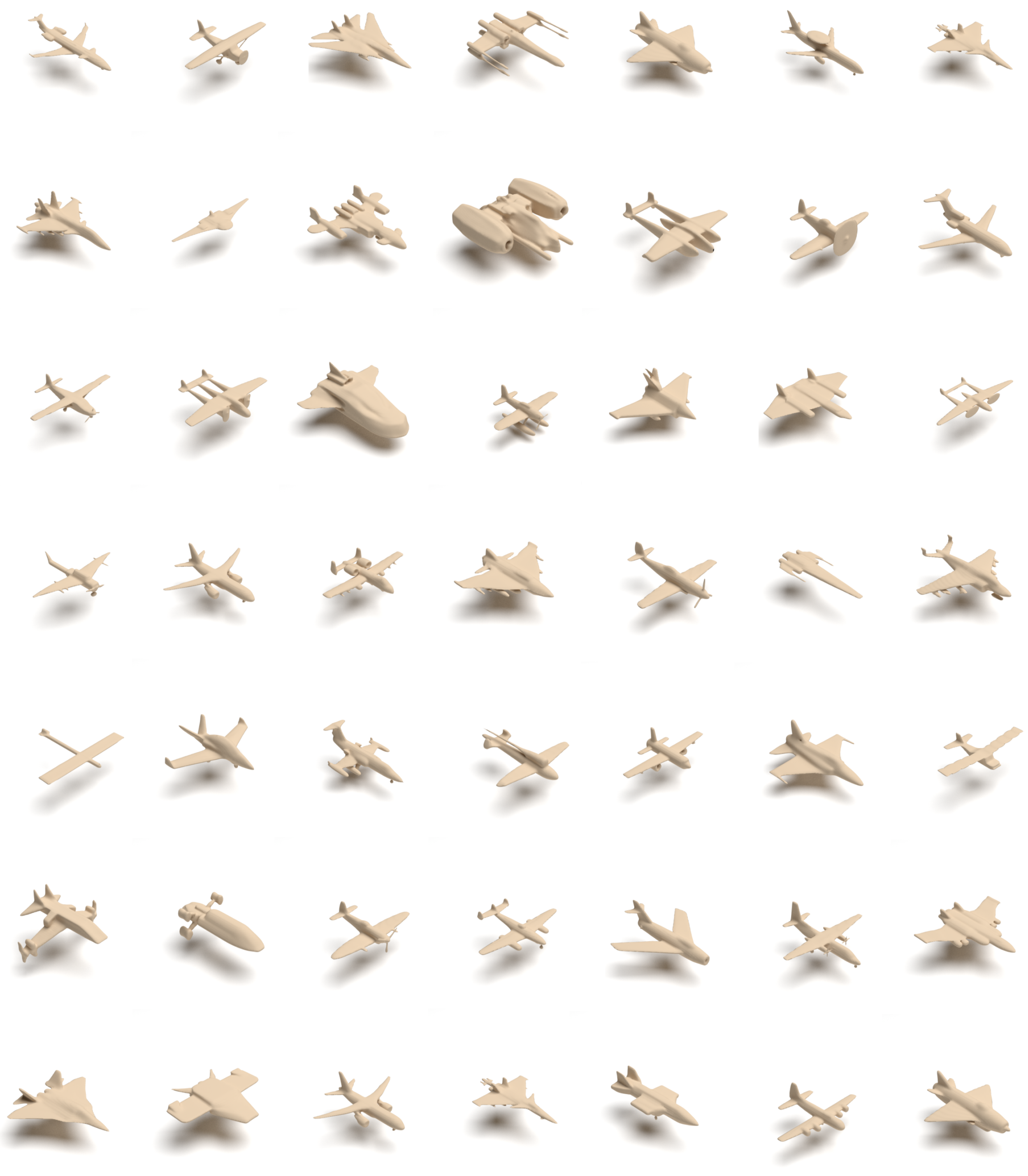


Figure 15. Gallery of airplanes generated by our method.

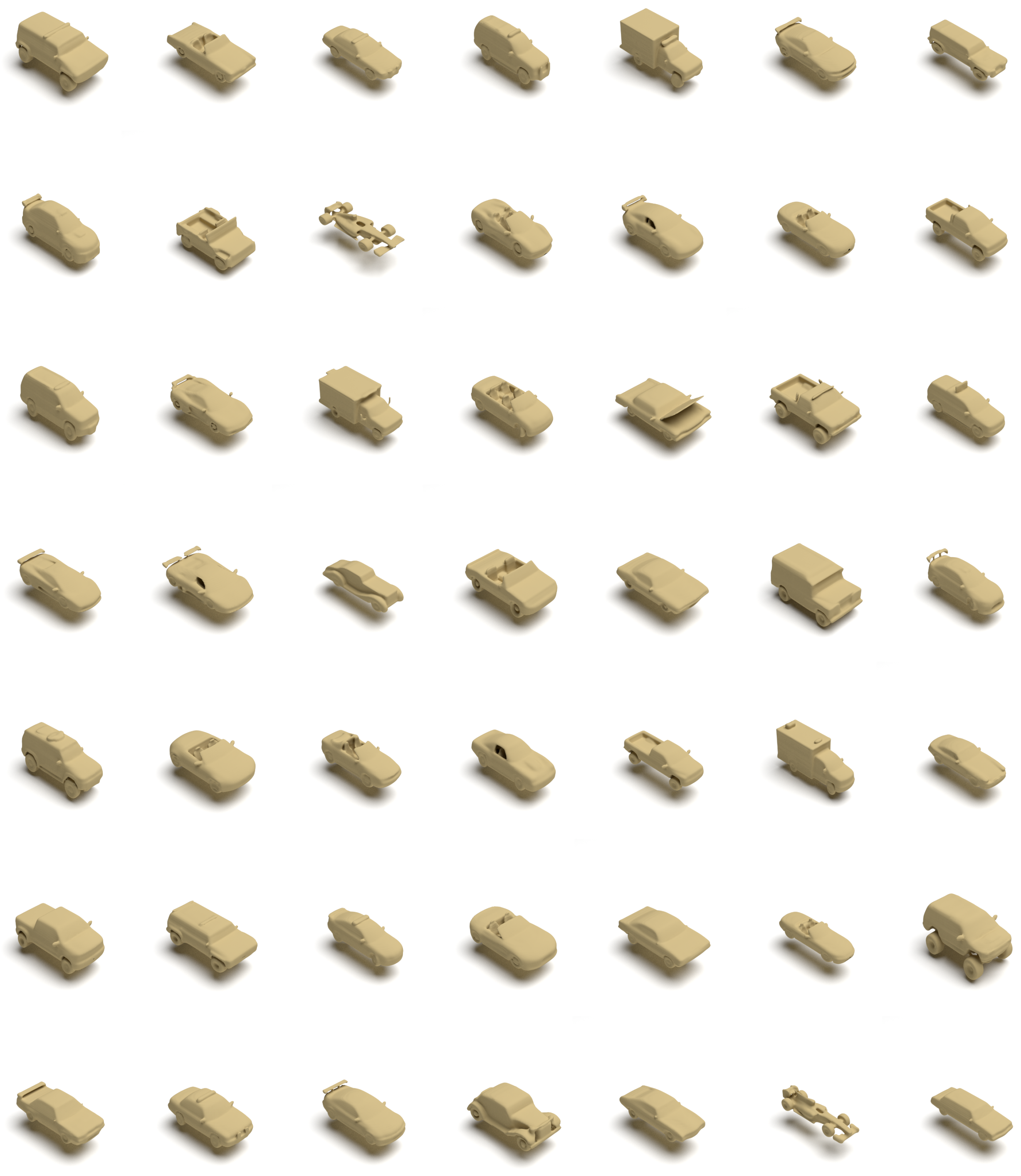


Figure 16. Gallery of cars generated by GPLD3D.

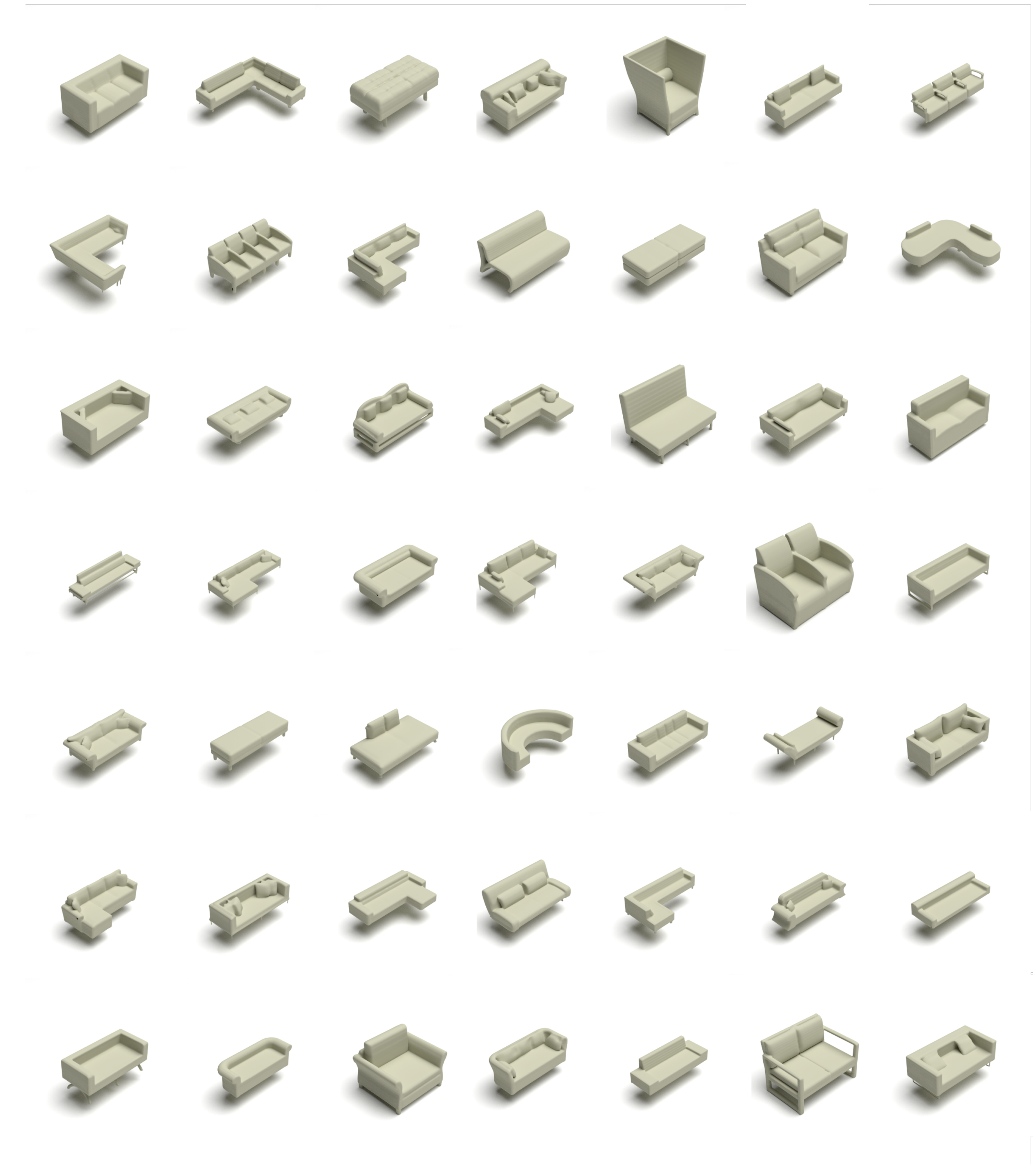


Figure 17. Gallery of sofas generated by GPLD3D.

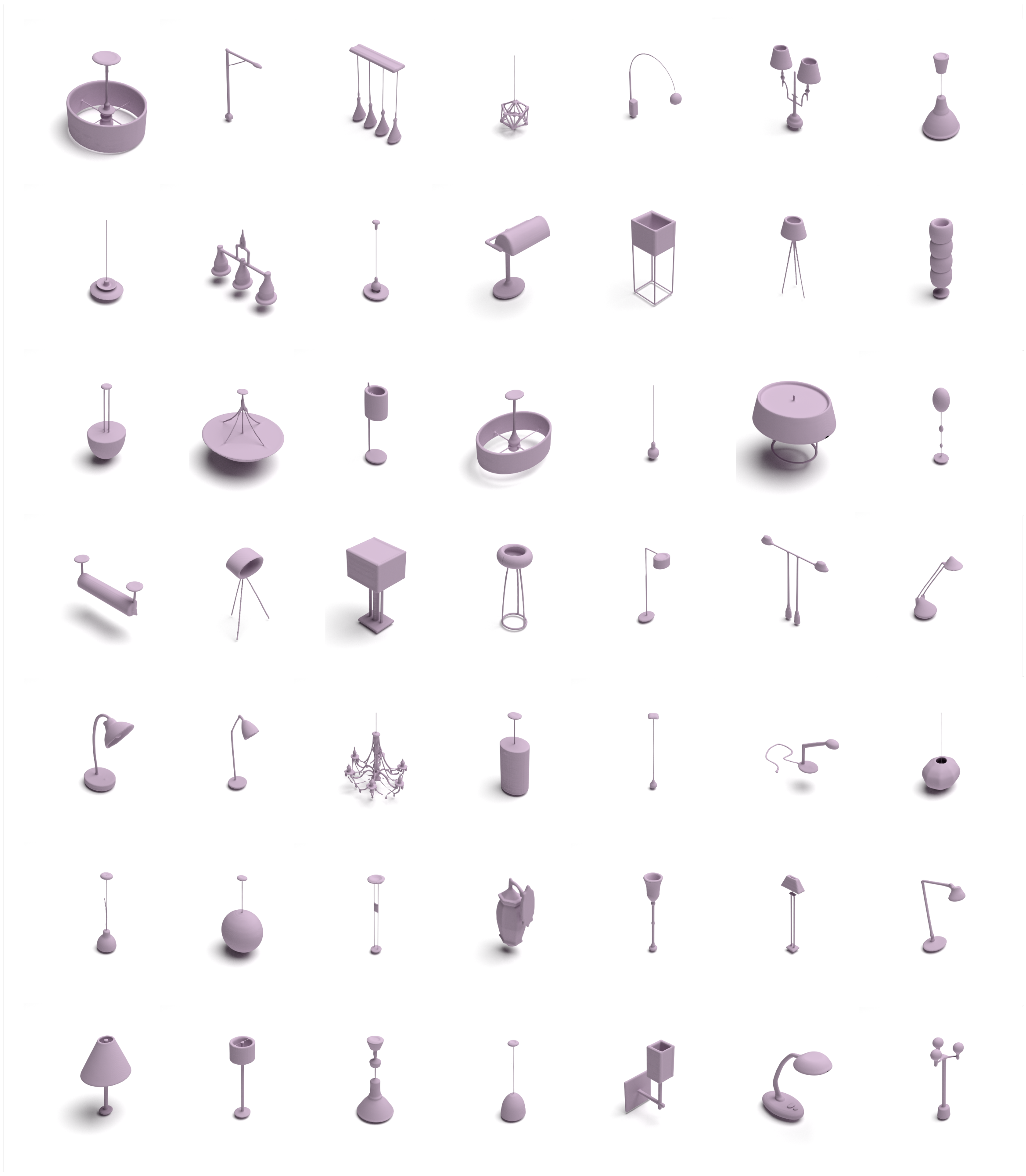


Figure 18. Gallery of lamps generated by GPLD3D.