A Diffusion-ReFinement Model for Sketch-to-Point Modeling (Supplementary Material)

Di Kong, Qiang Wang, and Yonggang Qi(✉)

Beijing University of Posts and Telecommunications, Beijing, China
{dikong,wanqqiang,qiyg}@bupt.edu.cn

1 Extended Derivations

We present the extended derivations of the training objective in main paper Equation (4). For the sake of brevity, we use the distribution with respect to the entire point cloud X^0. Since the points in a point cloud are independently sampled from a distribution, the probability of the whole point cloud is simply the product of the probability of each point:

$$q(X^1, ..., X^T|X^0) = \prod_{i=1}^{N} q(x_i^1, ..., x_i^T|x_i^0) \ ,$$

$$p_\theta(X^0, ..., X^T|z_s) = \prod_{i=1}^{N} p_\theta(x_i^0, ..., x_i^T|z_s) \ .$$

Having formulated the forward and reverse sampling processes for point clouds, we will formalize the training objective of the reverse sampling process for point cloud generation as follows.

The goal of training the reverse sampling process is to maximize the log-likelihood of the point cloud: $\mathbb{E}[logp_\theta(X^0)]$. However, since directly optimizing the exact log-likelihood is intractable, we instead *maximize* its variational lower bound:

$$\begin{aligned}\mathbb{E}[logp_\theta(X^0)] &\geq \mathbb{E}_q[log\frac{p_\theta(X^0, ..., X^T, z_s)}{q(X^1, ..., X^T|X^0)}] \\ &= \mathbb{E}_q[logp(X^T) + \sum_{t=1}^{T} log\frac{p_\theta(X^{t-1}|X^t, z_s)}{q(X^t|X^{t-1})}] \ .\end{aligned}$$

The above variational bound can be adapted into the training objective L to be *minimized*:

$$\begin{aligned}L(\theta) = \mathbb{E}_q[&\sum_{t=2}^{T} D_{KL}(q(X^{t-1}|X^t, X^0)||p_\theta(X^{t-1}|X^t, z_s)) \\ &- logp_\theta(X^0|X^1, z_s)] \ .\end{aligned}$$

And we present the detailed adaptation process as follows.

$$\begin{aligned}
\mathbb{E}_{q_{data}}[logp_\theta(X^0)] &= \int q_{data}(X^0)[log \int p_\theta(X^0, ..., X^T, z_s) dX^{1:T} dz_s] dX^0 \\
&\geq log \int q_{data}(X^0) p_\theta(X^{0:T}, z_s) dX^{1:T} dz_s dX^0 \quad (Jensen'sinequality) \\
&\geq \int q_{data}(X^0) q(X^{1:T}|X^0) log \frac{p_\theta(X^{0:T}, z_s)}{q(X^{1:T}|X^0)} dX^{1:T} dz_s dX^0 \quad (ELBO)
\end{aligned}$$

Note that $X^{1:T}$ and z_s are conditionally independent on X^0,

$$\begin{aligned}
&= \int q_{data}(X^0) q(X^{1:T}|X^0)[logp(X^T) + logp(z_s) \\
&\qquad + \sum_{t=1}^{T} logp_\theta(X^{t-1}|X^t, z_s) - \sum_{t}^{T} logq(X^t|X^{t-1})] dX^{0:T} dz_s \\
&= \int q_{data}(X^0) q(X^{1:T}|X^0)[logp(X^T) + logp(z_s) \\
&\qquad + \sum_{t=1}^{T} log \frac{p_\theta(X^{t-1}|X^t, z_s)}{q(X^t|X^{t-1})}] dX^{0:T} dz_s
\end{aligned}$$

Since $q(X^t|X^{t-1})$ is intractable, we rewrite it using Bayes' rule,

$$\begin{aligned}
&= \int q_{data}(X^0) q(X^{1:T}|X^0)[logp(X^T) + logp(z_s) \\
&\qquad + \sum_{t=2}^{T} log \frac{p_\theta(X^{t-1}|X^t, z_s)}{q(X^{t-1}|X^t, X^0)} \cdot \frac{q(X^{t-1}|X^0)}{X^t|X^0} + log \frac{p(X^0|X^1, z_s)}{q(X^1|X^0)}] dX^{0:T} dz_s \\
&= \int q_{data}(X^0) q(X^{1:T}|X^0)[log \frac{p(X^T)}{q(X^T|X^0)} \\
&\qquad + \sum_{t=2}^{T} log \frac{p_\theta(X^{t-1}|X^t, z_s)}{q(X^{t-1}|X^t, X^0)} + logp_\theta(X^0|X^1, z_s) + logp(z_s)] dX^{0:T} dz_s \\
&= \int q(X^{0:T})[log \frac{p(X^T)}{q(X^T|X^0)} + \sum_{t=2}^{T} log \frac{p_\theta(X^{t-1}|X^t, z_s)}{q(X^{t-1}|X^t, X^0)} \\
&\qquad + logp_\theta(X^0|X^1, z_s) + logp(z_s)] dX^{0:T} dz_s
\end{aligned}$$

On the right hand side, all the terms except $logp_\theta(X^0|X^1, z_s)$ can be rewritten into the form of the KL divergence. We show how to do it on one of the terms.

For the rest of the terms, it is similar.

$$\begin{aligned}
&\int q(X^{0:T}) log\frac{p_\theta(X^{t-1}|X^t, z_s)}{q(X^{t-1}|X^t, X^0)} dX^{0:T} dz_s \\
&= \int q(X^0, X^{t-1}, X^t) log\frac{p_\theta(X^{t-1}|X^t, z_s)}{q(X^{t-1}|X^t, X^0)} dX^{0,t-1,t} dz_s \\
&= \int q(X^{t-1}|X^0, X^t) q(X^0, X^t) log\frac{p_\theta(X^{t-1}|X^t, z_s)}{q(X^{t-1}|X^t, X^0)} dX^{0,t-1,t} dz_s \\
&= -\int q(X^0, X^t) D_{KL}(q(X^{t-1}|X^t, X^0)||p_\theta(X^{t-1}|X^t, z_s)) dX^{0,t} dz_s \\
&= -\mathbb{E}_{X^0, X^t, z_s}[D_{KL}(q(X^{t-1}|X^t, X^0)||p_\theta(X^{t-1}|X^t, z_s))]
\end{aligned}$$

Next, notice that $log\frac{p(X^T)}{q(X^T|X^0)}$ has no trainable parameters, so we can ignore it in the training objective. Finally, by negating the variational bound and decomposing the distribution, we obtain the training objective in Equation (4) as follows:

$$L(\theta) = \mathbb{E}_q[\sum_{t=2}^{T}\sum_{i=1}^{N} \underbrace{D_{KL}(q(x_i^{t-1}|x_i^t, x_i^0)||p_\theta(x_i^{t-1}|x_i^t, z_s))}_{L_i^{t-1}} - \sum_{i=1}^{N} \underbrace{log p_\theta(x_i^0|x_i^1, z_s)}_{L_i^0}] \ .$$

The training objective can be optimized efficiently since each of the terms on the right hand side is tractable and q is easy to sample from the forward diffusion process. Next, we elaborate on the terms to reveal how to compute the objective. $L_i^{t-1} : q(x_i^{t-1}|x_i^t, x_i^0)$ can be computed with the following closed-form Gaussian according to [3]:

$$q(x_i^{t-1}|x_i^t, x_i^0) = \mathcal{N}(x_i^{t-1}|\mu_t(x^t, x^0), \sigma_t^2 I) \ ,$$

where, using the notation $\alpha_t = 1 - \beta_t$, and $\overline{\alpha}_t = \prod_{i=1}^{t} \alpha_i$:

$$\begin{aligned}
\mu_t(x^t, x^0) =& \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t} x^0 + \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t} x^t \ , \\
\sigma_t^2 =& \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t \ .
\end{aligned}$$

$L_i^{t-1} : p_\theta(x_i^{t-1}|x_i^t, z_s)(t = 1, ..., T)$ are trainable Gaussians as described in main paper Equation (3). We further analyse L_i^{t-1}. Since both $q(x_i^{t-1}|x_i^t, x_i^0)$ and $p_\theta(x_i^{t-1}|x_i^t, z_s)$ are Gaussians, the term L_i^{t-1} can be expanded as:

$$L_i^{t-1} = \mathbb{E}_{x_i^0, x_i^t, z_s}[\frac{1}{2\beta_t}\| \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t} x_i^0 + \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t} x_i^t - \mu_\theta(x_i^t, t, z_s)\|^2] + C \ .$$

Evaluating L_i^{t-1} requires sampling x_i^t from $q(x^t)|x^0$. In principle, it can be done by sampling iteratively through the Markov chain. However, [3] showed

$q(x^t)|x^0$ is a Gaussian, thus allowing us to sample x_i^t efficiently without iterative sampling:

$$q(x^t|x^0) = \mathcal{N}(x^t; \sqrt{\overline{\alpha}_t}x^0, (1-\overline{\alpha}_t)I) \ .$$

Using the Gaussian above, we can parameterize x_i^t as $x_i^t(x_i^0, \epsilon) = \sqrt{\overline{\alpha}_t}x_i^0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}$, where $\epsilon \sim \mathcal{N}(0, I)$:

$$L_i^{t-1} = \mathbb{E}_{x_i^0, \epsilon, z_s}[\frac{1}{2\beta_t}\| \frac{1}{\sqrt{\alpha_t}}(x_i^t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon) - \mu_\theta(x_i^t, t, z_s)\|^2] + C \ .$$

The above equation reveals that $\mu_\theta(x_i^t, t)$ must predict $\frac{1}{\sqrt{\alpha_t}}(x_i^t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon)$ given x_i^t. Thus, $\mu_\theta(x_i^t, t)$ can be parameterized as:

$$\mu_\theta(x_i^t, t) = \frac{1}{\sqrt{\alpha_t}}(x_i^t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_\theta(x_i^t, t, z_s)) \ ,$$

where $\epsilon_\theta(x_i^t, t, z_s)$ is a function approximator (*i.e.*, neural network) intended to predict ϵ from x_i^t. Finally, L_i^{t-1} can be simplified as:

$$L_i^{t-1} = \mathbb{E}_{x_i^0, \epsilon, z_s}[\frac{\beta_t^2}{2\beta_t\alpha_t(1-\overline{\alpha}_t)}\|\epsilon - \epsilon_\theta(\sqrt{\overline{\alpha}_t}x_i^0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}, t, z_s)\|^2] + C \ .$$

To minimize L_i^{t-1}, we can only minimize $\mathbb{E}[\|\epsilon - \epsilon_\theta\|^2]$ because the coefficient $\frac{\beta_t^2}{2\beta_t\alpha_t(1-\overline{\alpha}_t)}$ is an invariant constant.

2 Implementation Details

In this work, we focus on sketch to point cloud reconstruction, which is a *conditional* reconstruction problem, because the Markov chain considered in our work generates points of a point cloud conditioned on some shape latent extracted from sketches. This conditional adaptation leads to significantly different training and sampling schemes compared to prior research on diffusion probabilistic models. And to better achieve cross-modal generation, we applied special designed training strategy.

Diffusion Process According to [3], there is a closed form expression for $q(x^t|x^0)$. We use the notation $\alpha_t = 1 - \beta_t$, and $\overline{\alpha}_t = \prod_{i=1}^t \alpha_i$. Then, we have $q(x^t|x^0) = \mathcal{N}(x^t; \sqrt{\overline{\alpha}_t}x^0, (1-\overline{\alpha}_t)I)$. Therefore, when T is large enough, $\overline{\alpha}_t$ goes to 0, and $q(x^T|x^0)$ becomes close to the ultimate distribution $q_{ultimate}(x^T)$. Note that x^t can be directly sampled through the following equation:

$$x^t = \sqrt{\overline{\alpha}_t}x^0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}, \text{ where } \boldsymbol{\epsilon} \text{ is a standard Gaussian noise} \ .$$

We emphasize that $q(x^t|x^{t-1})$can be seen as a one-to-one pointwise mapping as x^t can be sampled through the equation $x^t = \sqrt{1-\beta_t}\boldsymbol{x}^{t-1} + \beta_t\boldsymbol{\epsilon}$. Therefore, the order of points in x^0 is preserved in the diffusion process. However, it does not matter what kind of order we input the points in x^0. That is because when T is large enough, x^T will become a standard Gaussian distribution. And we are able to use the simple mean squared error because DPM naturally defines a one-to-one pointwise mapping between two consecutive point clouds in the diffusion process as shown in main paper Equation (1). Note that at each training step, we not only need to sample a point clouds x_i, but also a diffusion step t and a Gaussian noise ϵ.

Reverse Diffusion Kernel The reverse diffusion kernel in Equation (2) is parameterized by $\epsilon_\theta(x_i^t, t, z_s)$, as derived in Supplementary Section 1. The kernel takes the noisy point cloud x^t as input. We also add the diffusion step t, the shape feature z_s extracted from sketch to the denoising kernel. We implement it using a variant of PVCNN [4], which consists of a series of ConcatSquashConv1d layers [2] defined as:

$$h^{l+1} = CSC(h^l, t, z_s) = (W_1 h^l + b_1) \odot \sigma(W_2 c + b_2) + W_3 c \ .$$

where h^l is the input to the layer and h^{l+1} is the output. The input to the first layer is the 3D position of points x_i^t. $c = [\beta_t, sin(\beta_t), cos(\beta_t), z_s]$ is the context embedding vector, and σ denotes the sigmoid function. W_1, W_2, W_3, b_1 and b_2 are all trainable parameters.

ReFinement Network Although with the usage of the discriminator, the overfitting issue may happen when the discriminator only looks at clean samples, the diffusion process smoothens the data distribution [5], making the discriminator less likely to overfit. We visually and intuitively show the effectiveness of our discriminator design in main paper Section 4.

Convolutional and Attentional Block Our convolutional block consists of 2 convolutional layers. The convolutional block maps the coordinates of input point cloud to higher dimensions and propagates the sketch features along with the attentional block. And our attentional block is shown in Fig. 3 (b). It consists of 2 multi-head attention layers. The first attention layer performs attention between the output of the sketch-latent-shape encoder and the ground truth 3D point cloud, while the second performs self-attention. Sinusoidal positional encodings are added to the queries and keys at every attentional layer. For attentional layers at higher point cloud resolutions, the positions of the sketch shape latent are multiplied before the positional encoding to allow for easier alignment between the sketch latent and the distribution of the points in point cloud.

Experimental Details Both diffusion and reverse sampling processes have length equal to T. Using 1 NVIDIA V100 GPU, we set step iterations T to 200 in this paper. It takes an average of 0.45 seconds to generate a point cloud. Intuitively speaking, the *reverse* diffusion process is analogous to **MCMC** (Langevin dynamics) sampling procedures where β_t is the step size of the $(T - t + 1)$-th step from Equation (3). Since we normalize point clouds to unit variance and the coordinates of points roughly range from -2 to 2, we set the initial step size β_T to 0.05, slightly larger than $\frac{2-(-2)}{T} = \frac{4}{200}$, in order to ensure that the points can walk through possible regions of different shapes in early steps. To make the points concentrate in desired regions, $\beta_T, ..., \beta_1$ should be decaying and the last "step size" β_1 should be sufficiently small, so we set β_1 to 0.0001.

Dataset Details The ShapeNet-Synthetic dataset contains object renderings of 13 categories from ShapeNet [1]. Images of each object are rendered in 224x224 resolution. Following [8], we extract the edge map of each rendered image using canny edge detector, which dose not require 3D ground truth comparing to extracting contours from 3D shapes. We directly use the edge maps as synthetic sketches, and use this dataset to pretrain the model, under the same train/test/validation sets as the ratio 80%, 15%, 5% mentioned in main paper.

Regarding the number of points we use, for each shape in ShapeNetCorev2, we use farthest point sampling algorithm to uniformly sample 642 points from shape surface. All points are then centered and scaled. As shown in Figure 4, we believe the 642 points have preserved a good overall details and geometries of the objects. Compared with GAN, diffusion model is a conductive iterative generation process that takes more time and computational resources. To make this work more adaptable, we balanced the quality of the generation with the time required, and eventually chose 642 points per point cloud. And the way we sample 642 points from the generated meshes follows the processing approach of [9], which renders the object into points from a set of views and combines the renderings.

Each quadruplet in ShapeNet-Amateur dataset comprises one 3D shape and three free-hand sketches drawn from three azimuthal angles respectively. And we only used one of the three sketches with an azimuth of 45 degrees.

3 Additional Results

We show some additional results on ShapeNet-Amateur dataset in Fig. 1 and 2. And we also compare with OccNet [6] to display that our results are more sketch-aligned, shown in Fig. 3 and 4. In addition, Table 1 shows more comparison results with the latest methods.

Table 1. Comparison with two other baseline methods.

Category	AmaChair		AmaLamp		AmaMean		SynAirplane		SynCar		SynChair		SynLamp		SynMean	
	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD
Sketch2Model [8]	3.371	13.84	**5.928**	17.03	4.650	15.44	1.499	7.04	2.375	10.42	2.948	11.30	**5.472**	16.62	3.074	11.35
SRFHS [7]	**2.850**	12.88	6.273	17.82	**4.562**	15.35	1.561	6.87	2.367	16.69	**2.634**	12.56	6.033	16.25	3.149	13.09
Ours	3.250	**10.19**	6.152	**16.33**	4.701	**13.26**	**1.448**	**6.39**	**2.291**	**7.26**	2.865	**9.67**	5.564	**15.92**	**3.042**	**9.81**

References

1. Chang, A.X., Funkhouser, T.A., Guibas, L.J., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: Shapenet: An information-rich 3d model repository. CoRR **abs/1512.03012** (2015), http://arxiv.org/abs/1512.03012
2. Grathwohl, W., Chen, R.T., Bettencourt, J., Sutskever, I., Duvenaud, D.: Ffjord: Free-form continuous dynamics for scalable reversible generative models. arXiv preprint arXiv:1810.01367 (2018)
3. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems **33**, 6840–6851 (2020)
4. Liu, Z., Tang, H., Lin, Y., Han, S.: Point-voxel cnn for efficient 3d deep learning. Advances in Neural Information Processing Systems **32** (2019)
5. Lyu, S.: Interpretation and generalization of score matching. arXiv preprint arXiv:1205.2629 (2012)
6. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4460–4470 (2019)
7. Wang, J., Lin, J., Yu, Q., Liu, R., Chen, Y., Yu, S.X.: 3d shape reconstruction from free-hand sketches. arXiv preprint arXiv:2006.09694 (2020)
8. Zhang, S.H., Guo, Y.C., Gu, Q.W.: Sketch2model: View-aware 3d modeling from single free-hand sketches. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6012–6021 (2021)
9. Zhong, Y., Gryaditskaya, Y., Zhang, H., Song, Y.Z.: Deep sketch-based modeling: Tips and tricks. In: 2020 International Conference on 3D Vision (3DV). pp. 543–552. IEEE (2020)

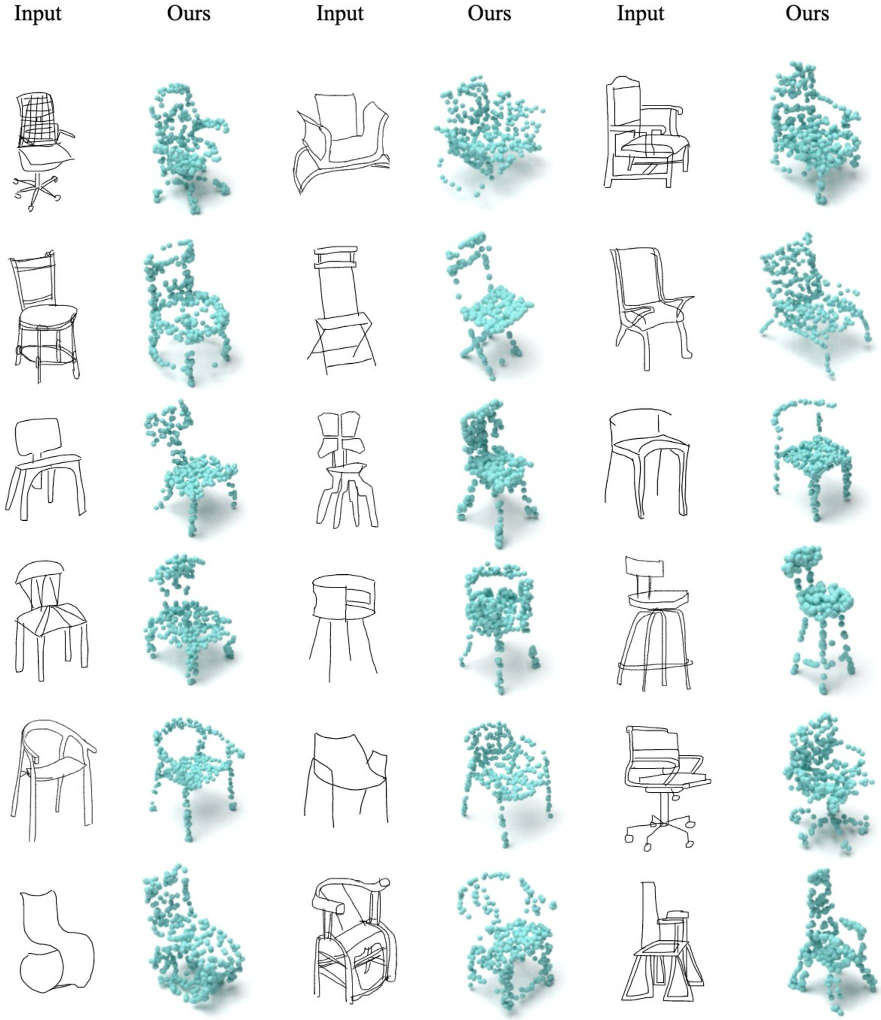

Fig. 1. Additional example results on ShapeNet-Amateur dataset.

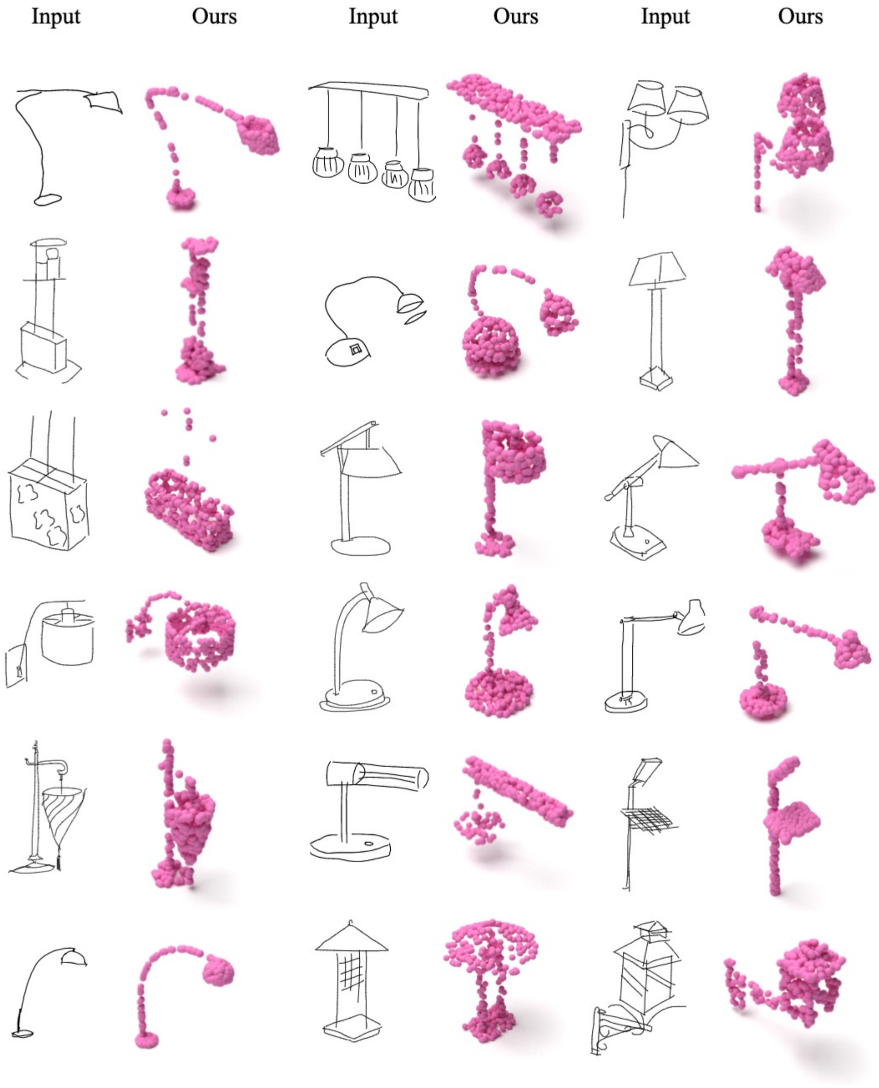

Fig. 2. Additional example results on ShapeNet-Amateur dataset.

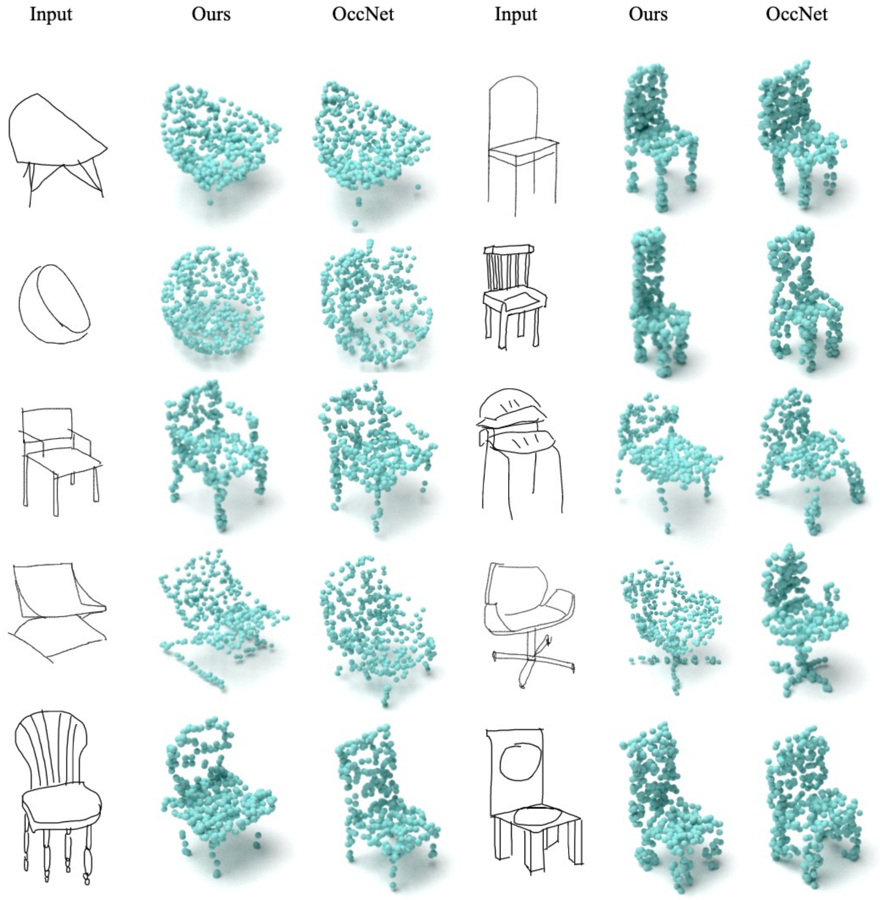

Fig. 3. Additional comparison results on ShapeNet-Amateur dataset.

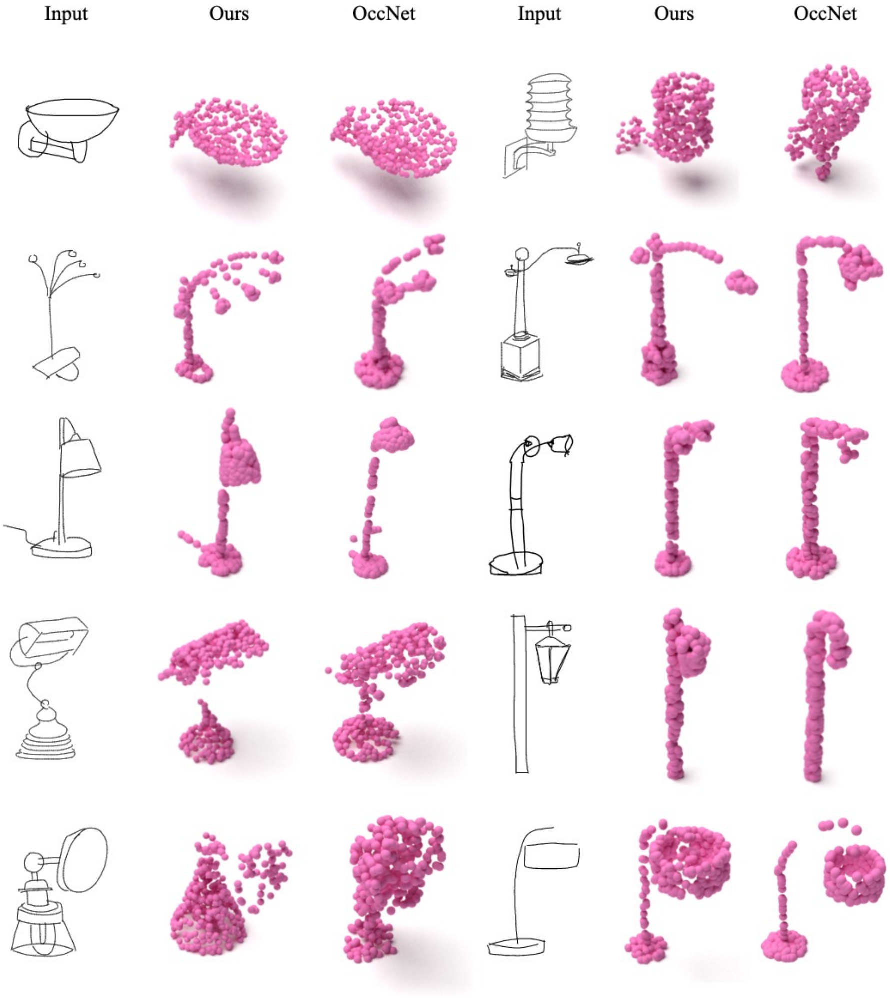

Fig. 4. Additional comparison results on ShapeNet-Amateur dataset.