Diffusion-SDF
Supplementary Material

Gene Chou
Princeton University
gchou@princeton.edu

Yuval Bahat
Princeton University
yb6751@princeton.edu

Felix Heide
Princeton University
fheide@princeton.edu

In this supplementary document, we present additional information and experiments in support of the main manuscript. We provide training and architecture details, formulations of evaluation metrics, and further analyses and visualizations.

Contents

1. Model and Training Details

In this section, we provide details for reproducibility, including model architecture, data processing, and implementation. We will also release code to facilitate experiments.

1.1. Model Architecture

As described in the main document, we use the SDF architecture from Chou *et al*. [5], a vanilla VAE [13] architecture, and the diffusion model from Aditya *et al*. [19, 23] with slight modifications to fit our data dimensions. We provide a full description below. We encourage readers to reference Fig. 2 in our main paper.

Joint SDF-VAE Model A raw input point cloud is passed through a PointNet encoder [18, 17] Ψ that consists of 5 ResNet [9] blocks, a fully-connected layer, and a UNet [20] with 4 2D convolutional layers, 3 2D transposed convolution layers, and a final convolution layer. The ResNet blocks have input and output dimensions of 128. The intermediate

fully-connected layer has output dimensions of 256. Between this fully-connected layer and the UNet, we project the point features to 3 2D planes to learn detailed spatial features. The UNet has hidden dimensions of $[32, 64, 128, 256, 128, 64, 32]$ and the final convolution layer has output dimensions of 256. This encoder takes in a point cloud with dimensions $(N, 3)$, and outputs 3 plane features each with dimensions $(256, 64, 64)$. We concatenate the plane features to obtain one feature with dimensions $(768, 64, 64)$. This becomes our input to the VAE Θ, which outputs a 1D latent embedding with dimensions $(768, 1)$. Our VAE has a 5-layer encoder and a 5-layer decoder. Each layer in the encoder has a 2D convolution layer with hidden dimensions 768, with kernel size 3, stride 2, padding 1, batch normalization [11], and ReLU activation. Each layer in the decoder has a transpose convolution layer with the same dimensions. Next, we project query points onto 2D planes, and perform interpolation to extract point features from the reconstructed plane features. We sum the point features from all 3 planes, and the point features have dimensions (N, D) where we set $D = 256$ and $N = 16,000$. The features are concatenated with the 3D coordinates (result has dimension $(N, D + 3)$) and fed into the SDF decoder Φ. The SDF decoder is a 9-layer MLP each with hidden dimensions 512 and a skip connection in the 4th layer. We use the ReLU activation after each layer except the last, and do not use any normalization layers.

Diffusion Model Our diffusion model has 6 layers, each with attention, fully-connected blocks, and layer normalization [1]. Each fully-connected block has hidden dimensions of 768 and we use dropout of 0.3. For self-attention, we use 768 for the dimension for the network that learns the key-value pairs. For cross-attention, we use 128, as we set the output dimension of our custom encoder Υ to 128. We also employ a sinusoidal positional embedding followed by a 2-layer MLP for the timestep t. The input vector is concatenated with the embedded timestep before being passed into the diffusion model.

Custom Encoder for Conditioning For our custom encoder for extracting shape features for conditioning our diffusion model, we use the same PointNet in GenSDF [5, 17] for point clouds, and a pretrained ResNet 18 [9] for images. Our ResNet 18 and its pretrained weights are directly loaded from the PyTorch [16] *torchvision models* library.

For conditioning on point clouds, we pass all points through our encoder Υ and obtain point features without performing pooling or interpolations. We set the output dimensions to 128. For conditioning on images, we add a final fully connected layer with output dimensions 256 to the ResNet.

1.2. Data Processing

Preprocessing We recenter and normalize all objects in the Acronym [8] dataset using the normalized object coordinate system (NOCS) [22]. We center each object within a 3D cube uniformly scale each object such that the diagonal of its tight bounding box has a length of 1. We set our cube to have coordinates ranging from $(-1, -1, -1)$ to $(1, 1, 1)$.

To obtain query and signed distance value pairs for training, we follow Chou et al. [5] for preprocessing. From each mesh we sample 235,000 points on the surface. These points form a point cloud $P = \{p_i \in \mathbb{R}^3\}_{i=1}^{235000}$ that describe the surface and have signed distances of 0. Then, for each point p_i, we sample two Gaussian distributions with mean 0 and standard deviation 0.005 and 0.0005, respectively. We add these distributions to p_i to obtain two query points per surface point. In addition to these points near the surface, we store all points from a 3D grid ranging from $(-1, -1, -1)$ to $(1, 1, 1)$ with resolution $(128, 128, 128)$. With the mesh available, we can calculate ground truth signed distance values for supervised training and we store all points and corresponding signed distances. During one training step of one mesh, we randomly sample 1,024 surface points for generating shape features, and sample 16,000 query points for learned signed distance values. Of the 16,000 points, we use 30% uniformly sampled from the grid and the rest from the near surface points.

Partial Point Clouds To obtain partial point clouds, we follow Yu *et al*. [26] to randomly select a viewpoint and remove the n furthest points from the viewpoint to obtain a partial point cloud. In our experiments, we sample 128 points from a full point cloud, then remove 50% of the furthest points. During training, we perform this cropping online in each iteration and once per point cloud during testing.

YCB Point Clouds Unlike meshes, YCB [3] point clouds do not have a defined surface boundary and we cannot calculate ground truth signed distance values. We follow [14] and use points on the point cloud as proxies for approximating the surface boundary. For each input point cloud, we randomly sample 30,000 points and obtain $P = \{p_i \in \mathbb{R}^3\}_{i=1}^{30000}$. Then for each point p_i, we sample 20 Gaussian distributions each with mean 0 and standard deviation σ, where σ is the distance between p_i and its 50-th nearest neighbor. We add each distribution to p_i to obtain 20 query points near the surface. Next, to ensure the model does not overfit to generated queries that only take up a small spatial proportion of the cube, we uniformly

sample additional points within the cube with each dimension ranging from -1 to 1. The ratio of query points near the surface and uniformly from the grid is 3:1. We then store the nearest neighbor p to each query point x by finding the shortest distance between x and $p \in P$. During training, we sample 5,000 points from the point cloud and 5,000 query points each iteration. During testing, we sample 5,000 points from the point cloud for generation.

2D ShapeNet Images We use the 2D images rendered by Choy *et al*. [6] performed on ShapeNet [4] and pair them with our preprocessed Acronym [8] data. Acronym is a subset of ShapeNet so most meshes are paired with the renders under the same filename, and we omit the meshes that do not have corresponding renders. There are 23 renders per mesh, and we randomly pick one during each training iteration.

1.3. Implementation

We use PyTorch [16] and train all models with the ADAM [12] optimizer. For our joint SDF-VAE model, we set constant learning rates to 1×10^{-4}. For our diffusion model and custom encoder, we set constant learning rates to 1×10^{-5}. During end-to-end training, we use the same learning rates and fine-tune all modules (i.e., we do not freeze any parameters). We use a NVIDIA A100 GPU with 40GB of GPU memory for all experiments. On single categories (chair and couch), training the joint SDF-VAE model takes 3 days and training the diffusion model takes 6 hours for unconditional generation and 1 day for conditional generation. We train the two modules end-to-end for 1-2 more days. On our multi-category dataset (106 classes), we train for twice as long.

2. Metrics

For unconditional generation, we follow Yang *et al*. [25] and use minimum matching distance (MMD), coverage (COV), and 1-nearest neighbor accuracy (1-NNA). Let S_g be the set of generated meshes and S_r the set of reference meshes. For evaluation, we calculate the distance between point clouds, so from each mesh we sample 2,048 points on its surface. We generate the same number of samples as the reference set, i.e., $|S_g| = |S_r|$. For conditional generation (shape completion), we follow Wu *et al*. [24] and evaluate MMD, total mutual difference (TMD), and unidirectional Hausdorff distance (UHD). From each reference point cloud $P_i \in S_r$, we obtain a partial point cloud $p_i \subset P_i$. From p_i, we generate $k = 10$ samples, and denote Φ_{ij} the j-th sampled SDF. Then, we obtain a mesh c_{ij} from Φ_{ij} by running marching cubes. Thus, $S_g = \{\sum_{i=1}^{|S_r|} \sum_{j=1}^{k} c_{ij}\}$ and $|S_g| = k|S_r|$. For all metrics except UHD, we use Chamfer Distance (CD) [18] as the distance measure.

Minimum Matching Distance (MMD) For each point cloud in S_r, we compute its distance to its nearest neighbor in $|S_g|$. MMD measures quality with a score that is low when the generated set consists of point clouds close to those in the reference set.

$$\text{MMD}\left(S_g, S_r\right) = \frac{1}{|S_r|} \sum_{Y \in S_r} \min_{c \in S_g} \text{CD}(c, Y),$$

where CD is the Chamfer Distance.

Coverage (COV) For each point cloud in $|S_r|$, we find its nearest neighbor in $|S_g|$. COV measures the fraction of point clouds in $|S_r|$ that are mapped to a *unique* nearest neighbor in $|S_g|$. In other words, if there is mode collapse, then the reference point clouds will map to a small set of nearest neighbors, leading to a low COV score.

$$\text{COV}\left(S_g, S_r\right) = \frac{|\{\arg\min_{Y \in S_r} \text{CD}(c, Y) \mid c \in S_g\}|}{|S_r|}.$$

1-Nearest Neighbor Accuracy (1-NNA) 1-NNA measures the similarity between two shape distributions rather than marginal point distributions. Intuitively, a 1-NN classifier classifies each sample as coming from $|S_r|$ or $|S_g|$, and if the generated and reference sets are indistinguishable, the classifier accuracy should be close to 50%. Let $S_{-c} = S_r \cup S_g - \{c\}$ and N_c the nearest neighbor of c in S_{-c}.

$$\text{1-NNA}\ \left(S_g, S_r\right) = \frac{\sum_{c \in S_g} \mathbb{I}\left[N_c \in S_g\right] + \sum_{Y \in S_r} \mathbb{I}\left[N_Y \in S_r\right]}{|S_g| + |S_r|},$$

where $\mathbb{I}$ is the indicator function.

Total Mutual Difference (TMD) TMD is defined as the average of all pairwise distances between all generated samples given a conditional input, that is

$$\text{TMD}(S_g) = \frac{1}{|S_g|} \sum_{i=1}^{|S_g|} \left(\sum_{j=1}^{k} \frac{1}{k-1} \sum_{1 \leq l \leq k, l \neq j} \text{CD}\left(c_{ij}, c_{il}\right) \right).$$

Unidirectional Hausdorff distance (UHD) UHD is the average unidirectional Hausdorff distance from the conditioned input shape to each of the generated shapes, that is

$$\text{UHD}(S_g, S_r) = \frac{1}{|S_g|} \sum_{i=1}^{|S_g|} \left(\frac{1}{k} \sum_{j=1}^{k} \text{HL}\left(p_i, c_{ij}\right) \right),$$

where HL is the unidirectional Hausdorff distance.

Consistency (CONS) We evaluate all points in the input partial point cloud using the generated SDF and take the average of the predicted signed distance values. If the points are present on the reconstructed surface, then by definition, the values of each point are close to 0. This metric can only be used for SDFs so we use it for our ablation study and for filtering generations before marching cubes.

$$\text{CONS} = \frac{1}{|S_g|} \sum_{i=1}^{|S_g|} \left(\frac{1}{k} \sum_{j=1}^{k} ||\Phi_{ij}(p_i)||_1 \right),$$

where Φ_{ij} is the j-th generated SDF guided by partial point cloud p_i, and $||\cdot||_1$ denotes the $\mathcal{L}_1$ loss between the predicted signed distance values of points in p_i and 0.

3. Additional Analysis

3.1. Completion of Dense, Partial Point Clouds

In our main document we primarily experiment with sparse, partial point clouds. We follow [26] for creating partial point clouds: fix a viewpoint, and remove the n furthest points. In our case, we take a full point cloud with 128 points and crop 50% of the points. Here, we train and test with dense, partial point clouds on the Couch data split. Specifically, we take a full point cloud with 2048 points and crop 50%. We show qualitative visualizations in Fig. 1 and quantitative results in Tab. 1. Under this setting, our method remains capable of generating realistic, multi-modal outputs. Unsurprisingly, as measured by our metrics, diversity decreases and fidelity increases.

Figure 1. Generations guided by dense, partial point clouds (1024 points, 50% cropped). We render the point cloud from a viewing angle that is indicative of its overall shape but note that most points on the back of the point cloud are missing, see text for details on the cropping approach. Our method is capable of completing realistic, multi-modal outputs.

3.2. Modulation Comparisons

Our modulation module is the cornerstone of our SDF representation. The modulated latent vectors are used as training data for the diffusion model, so they are the bottleneck of the quality of generated meshes. Here, we show quantitative results and visualizations of different modulations.

Table 1. Quantitative results of conditional generations from sparse and dense partial point clouds, trained on the Couch split. We use 64 points for the former and 1024 points for the latter. Guiding with dense partial point clouds leads to lower diversity and higher diversity. Values are scaled up by 10^2.

Condition	TMD (↑)	CONS (↓)
Sparse (64 pts)	13.53	1.967
Dense (1024 pts)	10.42	1.259

Table 2. Average Chamfer Distance (CD) between modulation representations of SDFs and ground truth. ↑ means higher is better and ↓ means lower is better. All values are scaled up by 10^2.

	Couch	Multi-class
SIREN + meta-learning [7]	0.763	5.666
Auto-decoder [15]	0.557	17.83
Ours ($\sigma = 1.0$)	0.828	0.942
Ours ($\sigma = 0.5$)	0.108	0.621
Ours ($\sigma = 0.25$)	**0.104**	**0.607**

Dupont *et al*. [7] use SIREN and meta-learning to learn a linear mapping between latent vectors and a base network. Specifically, they implement SIREN [21] as their base network, and compute a second-order gradient to initialize a latent vector in 3 steps [2]. On single categories, this method performs adequately, but completely fails to represent detailed shapes when we increase the number of categories. We also experiment with auto-decoders [15]. At the start of training, we initialize a latent vector for each training data point. The latent vectors are linearly mapped to a base SDF network, represented by a standard MLP with ReLU activations, and the model updates the latent vectors accordingly. These two methods rely on linear mappings between *discrete* latent vectors and a base SDF network, and the distribution of shapes they can represent is limited. It is also difficult to regularize discrete latent vectors. In contrast, our method learns a regularized latent space and a shape prior and it is capable of representing over 100 categories with fine details. The experiments in Tab. 2 and Fig. 2 validate this.

In Sec. 4 of our main paper, we mention that we use a KL-loss to enforce the target latent distribution to be a zero-mean Gaussian with standard deviation 0.25. In *Ours ($\sigma = 1.0$)* and *Ours ($\sigma = 0.5$)* of Tab. 2, we experiment with standard deviations 1.0 and 0.5, respectively. Increasing σ can lead to an overly spread-out distribution that cannot encode shapes well, although we speculate that on datasets substantially larger than the ones we use, σ could be increased. But for our dataset, even when using 7148 meshes as explained in Sec. 5.4 of the main paper, $\sigma = 0.25$ was optimal. Apart from representation, our current choice of 0.25 allows the diffusion model to learn from a more compact distribution, which improved convergence speeds and training stability.

Additionally, in Fig. 3, we further validate the need for a modulation representation. We train a diffusion model to *overfit* a single SDF represented by an 8-layer MLP, without any modulations or compression. The goal is to confirm whether the diffusion model can learn using SDFs as data. Since we are overfitting on one object, the goal of the diffusion model is to reconstruct the object rather than generate new samples. However, directly training on SDFs is difficult because small noise in the SDF network can lead to drastically different outputs after running marching cubes. We show outputs of the same model trained twice. Training losses are equally low but the implicitly represented geometry varies because the diffusion model learns to reverse the weights of the neural SDF, which do not carry explicit geometric information. Furthermore, the diffusion model does not converge when trained on more than one SDF because it cannot find a distribution that models the individual SDFs.

3.3. Conditional Dropout

To increase diversity and prevent overfitting, we follow Ho *et al*. [10]; every training iteration, with a certain probability we use a zero-mask instead of the shape feature as condition. In practice, we use the zero-mask with probability 80%. During sampling, Ho *et al*. [10] use the following linear combination of conditional and unconditional estimates:

$$\tilde{\boldsymbol{\epsilon}}_\theta\left(\mathbf{z}_\lambda, \mathbf{c}\right) = (1+w)\boldsymbol{\epsilon}_\theta\left(\mathbf{z}_\lambda, \mathbf{c}\right) - w\boldsymbol{\epsilon}_\theta\left(\mathbf{z}_\lambda\right)$$

where $\boldsymbol{\epsilon}_\theta$ is the diffusion model, $\boldsymbol{c}$ is the condition, and $\boldsymbol{z}$ is the noisy data. Ho *et al*. find that increasing ω, the ratio of guidance strength, leads to higher fidelity but lower diversity, as measured by Inception Score and FID, respectively, of

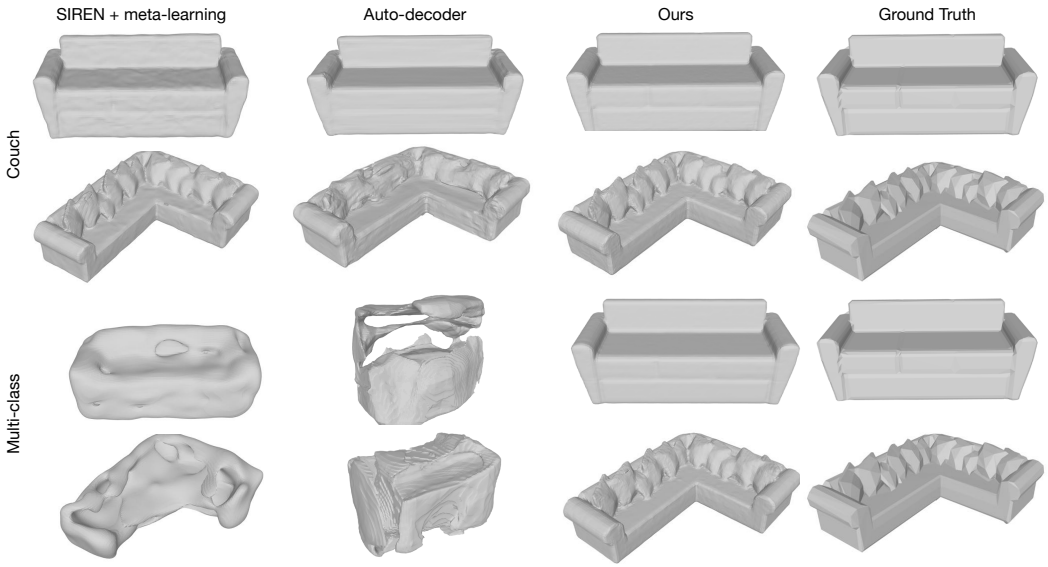

Figure 2. SDF reconstructions of different modulation methods. The first two rows are trained on the Couch data split. SIREN+meta-learning and auto-decoder methods perform well although the latter loses details such as the pillows. The bottom two rows are trained on the multi-class split and the same two couches are reconstructed (the couches are also in the multi-class split). Only our method is capable of reconstructing with detail due to our regularized latent space and generalizable shape prior.

Figure 3. A naive approach to training diffusion models on SDFs is to directly learn to reverse the weights of SDFs without modulations or compression. Here, a diffusion model is trained to *overfit* on a single SDF represented by an 8-layer MLP. Directly training on SDFs is difficult because small noise in the SDF network can lead to drastically different outputs after running marching cubes. We show outputs of the same model trained twice. Training losses are equally low but the implicitly represented geometry varies drastically. This motivates the creation of a modulation representation.

generated 2D images in their case [10]. In our ablation study (Sec. 5.4 in main paper), we also showed results of tuning ω and arrived at similar results; increasing ω led to lower CONS (higher fidelity) but lower TMD (lower diversity). Depending on the use case, one can adjust this hyperparameter to determine the tradeoff between diversity and fidelity without any retraining or fine-tuning.

3.4. Filtering using the CONS Metric

In Tab. 3 we report the quantitative values after training on the Multi-Class and Couch datasets with and without filtering. To filter, we sample the maximum number of meshes that can fit into one sampling batch, 30 in our case, and keep 10 meshes with the highest consistencies (i.e., smallest average signed distance values). Specifically, we evaluate all points in the input partial point cloud using the generated SDF and take the average of the predicted signed distance values. If the points are to be present on the reconstructed surface, then by definition, the values of each point are close to 0.

We do not use the CONS filter during training, only for conditional generations. We note that our model prioritizes generation quality and diversity, at the cost of fidelity. Our CONS filter provides a straightforward way to filter inconsistent generations using the definition of SDFs without incurring additional sampling time and does not inflate the MMD (quality) and TMD (diversity) metrics.

Table 3. Metrics for generating shapes after training on the Multi-Class and Couch datasets, with and without filtering. ↑ means higher is better and ↓ means lower is better. All values are scaled up by 10^2.

Category	MMD (↓)	TMD (↑)	UHD (↓)
Multi-class	0.035	35.28	24.20
Multi-class (no filt)	0.035	20.11	14.86
Couch	0.041	13.53	1.967
Couch (no filt)	0.041	17.06	3.545

Table 4. Average CD of reconstructing training data. One is trained on only the Couch dataset (366 meshes) and one is trained on all categories (7148 meshes). The model architecture, number of parameters, and training method are exactly the same. ↓ means lower is better. All values are scaled up by 10^3.

Training Data	# meshes	CD of couches (↓)	CD of all meshes (↓)
Couch	366	1.04	-
All classes	7148	0.87	0.92

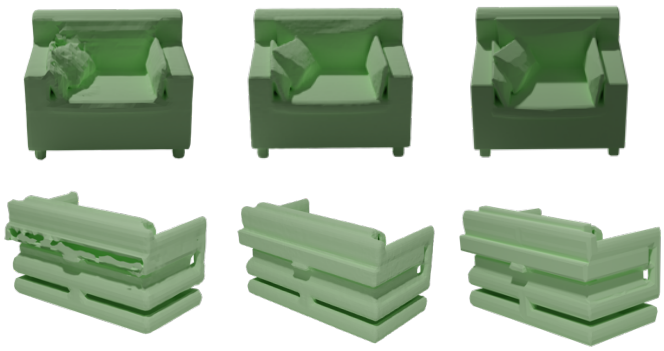

Figure 4. From left to right: reconstructions of training only on the Couch dataset, training on all 7148 meshes, ground truth. We run marching cubes the mesh the reconstructed SDFs at the same resolution (128^3). Generally both models reconstruct detail but for challenging cases the model trained on more data performs better.

3.5. Discussion on Scalability

We investigate whether our VAE poses a bottleneck to learning large datasets. The single-category experiments (Chair, Couch) are performed on 558 and 366 meshes, respectively. Our multi-class split contains 4230 meshes. As such, without adjusting the architecture or number of parameters, our method scales without degradation in quality or creating artifacts. To further validate scalability, we experiment with 90% of the entire Acronym dataset, 7148 meshes in total. We show a quantitative comparison of the Chamfer Distance (CD) of reconstructions between training on the Couch category and the large dataset in Tab. 4. For the latter, the CD of reconstructing couches is *lower* than training on the single Couch category. Furthermore, the CD of reconstructing all categories remains low. This validates that our approach scales gracefully, learns better when we introduce more training data, and generalizes to out-of-distribution shapes as many categories have very few data (just 1-10 meshes) compared to the larger classes (300-500 meshes).

Here in Fig. 4, we show that for challenging cases (such as high frequency and thin structures) where training only on the Couch dataset fails, the scaled model is capable of reconstructing detail. Our method also generalizes to out-of-distribution categories. In Fig. 5, we show shapes from a few categories that have fewer than 10 objects. We also show failure cases in the bottom row, mostly as a result of thin structures. However, we did not observe any pattern that led to the successful or unsuccessful reconstruction of thin structures. For instance, in Fig. 5, both cages (top and bottom row) have similar geometries, but reconstruction quality differs significantly. We leave this to future work.

Although we did not encounter issues with scalability in our experiments, we would be interested in expanding our 1D bottleneck to 2D in future work. This would preserve the spatial information of the feature planes, and could potentially require fewer parameters or fit to even larger datasets.

Figure 5. (Top) Reconstructions of out-of-distribution categories after training on all 7148 meshes. Each category has fewer than 10 meshes. From left to right: Backpack, Bird, Cage, Cow. The model captures detailed geometries. (Bottom) Failure cases mostly as a result of thin structures. From left to right: Fan, Bag, Cage, Glasses.

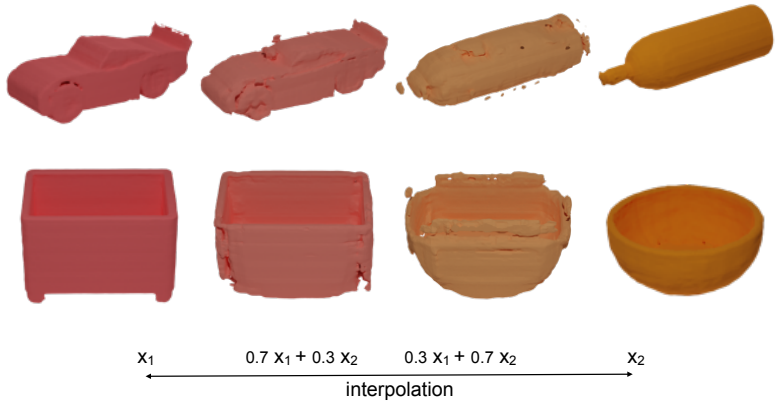

Figure 6. Failure cases of interpolation. (Top) We do not account for semantically meaningless geometries (e.g., interpolating between a car and a bottle). (Bottom) The two samples are far apart in the latent space and interpolations contain artifacts.

4. Limitations

The main limitation of our work is the lack of constraints on the specific shapes the method generates. As shown in Fig. 8 and Fig. 9, conditional generations may not be fully consistent with the condition. Our current solution is to utilize a CONS filter but to solve this issue, we could map the latents of partial shapes to those of their complete shapes during training to enforce consistency.

Additionally, the latent space is not heavily regularized so interpolations between latents may not be semantically meaningful. We show examples in Fig. 6. In the top row, the interpolation between a car and a bottle results in a semantically meaningless shape. This could be solved by introducing semantic information such as through text. In the bottom row, the two samples are far apart in the latent space, leading to artifacts in interpolations. This is somewhat expected because during training, our model samples from a region in the latent space and does not account for interpolating between two latents far from each other. In future work, we could set a perception loss for such an application.

5. Additional Unconditional Generations

We present additional visualizations for unconditional generation in Fig. 7. These results further validate that the proposed method is capable of producing clean meshes with thin structures and diverse geometries.

Figure 7. Additional samples from unconditional generation using the proposed method. Our method produces clean meshes with thin structures and diverse geometries.

6. Additional Conditional Generations

We show additional generations guided by sparse, partial point clouds in Fig. 8 and Fig. 9. We also show here show how filtered-generated SDFs can guide the output of the method at test time. When we filter based on a strict threshold (i.e., low CONS value), we obtain more deterministic but consistent outputs. As we increase the threshold (i.e., higher CONS value), we obtain diverse outputs that match the overall shape of the input point cloud. In the first column of our generated outputs (Fig. 8 and Fig. 9), nearly all points of the input are located on the mesh surface, while we obtain less precise but plausible outputs in the following columns. The threshold for filtering allows users to generate samples based with different properties at test time.

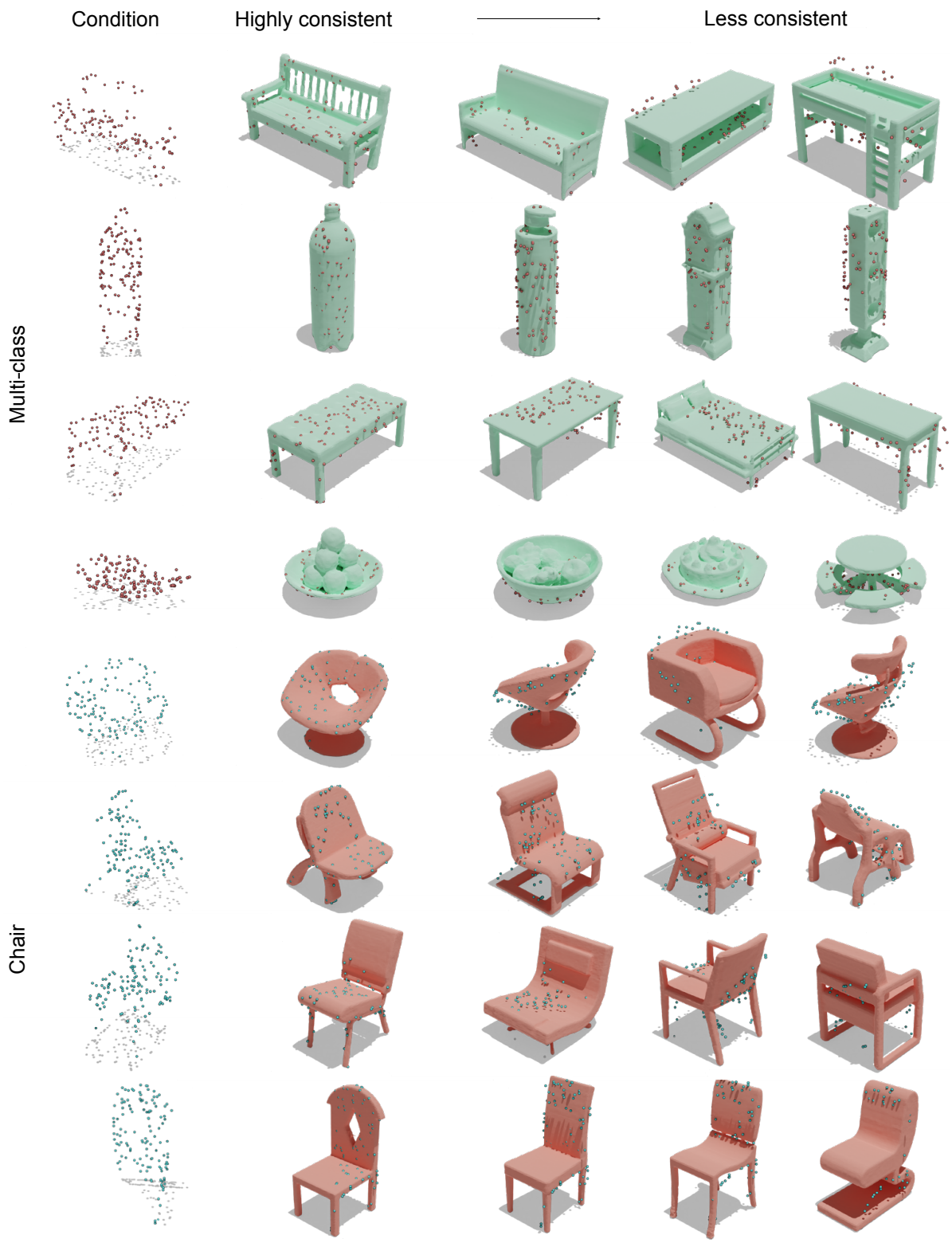

Figure 8. Generations guided by sparse, partial point clouds. Varying a threshold for filtering using our CONS metric, we can adjust the behavior of the method. A stricter threshold produces more consistent but deterministic outputs, while a looser threshold produces more diverse but less faithful outputs. For each row, we overlay the same point cloud onto all meshes although we render them at different viewing angles to show diversity of outputs.

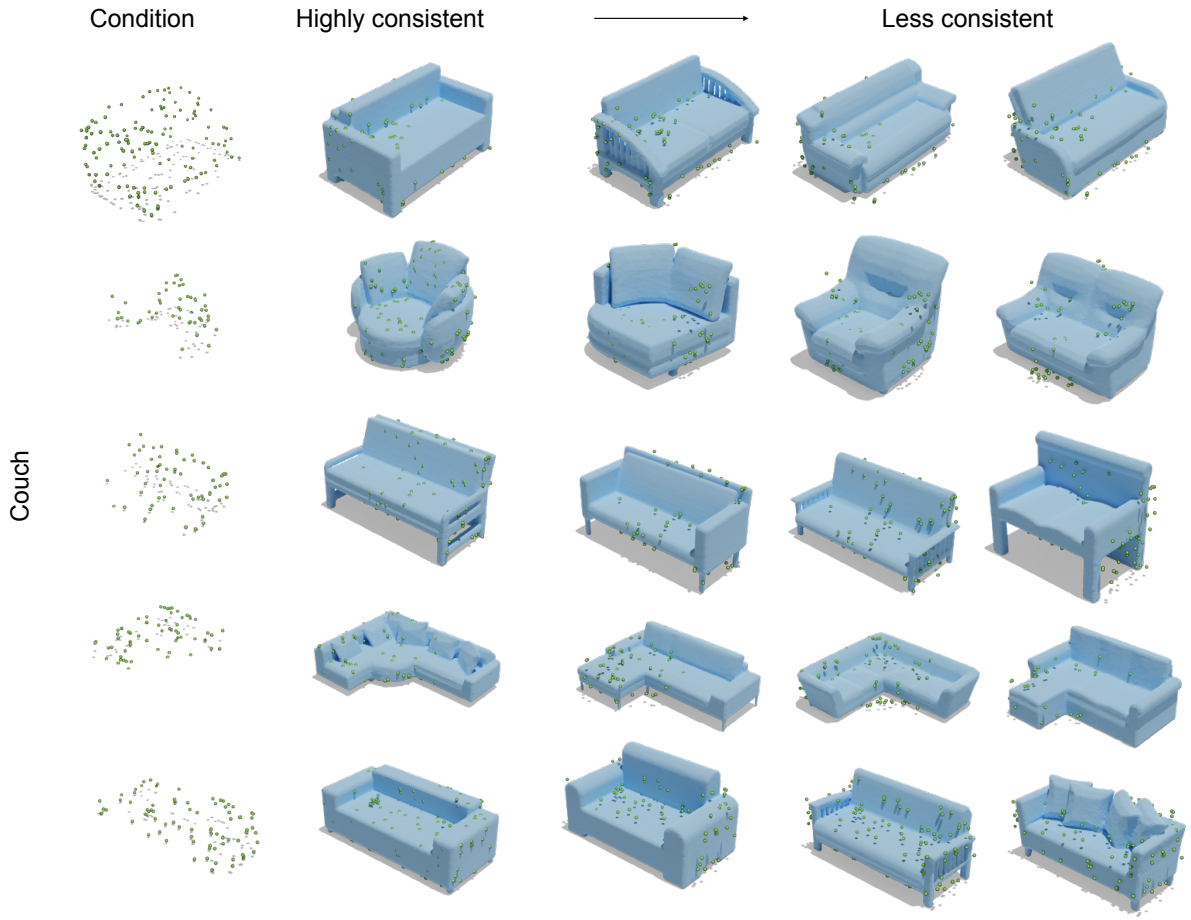

Figure 9. Additional generations guided by sparse, partial point clouds. Varying a threshold for filtering using our CONS metric, we can adjust the behavior of the method. A stricter threshold produces more consistent but deterministic outputs, while a looser threshold produces more diverse but less faithful outputs. For each row, we overlay the same point cloud onto all meshes although we render them at different viewing angles to show diversity of outputs.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 2

[2] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 5

[3] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015. 2

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

[5] Gene Chou, Ilya Chugunov, and Felix Heide. Gensdf: Two-stage learning of generalizable signed distance functions. In *Proc. of Neural Information Processing Systems (NeurIPS)*, 2022. 1, 2

[6] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 3

[7] Emilien Dupont, Hyunjik Kim, SM Ali Eslami, Danilo Jimenez Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. In *International Conference on Machine Learning*, pages 5694–5725. PMLR, 2022. 5

[8] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. ACRONYM: A large-scale grasp dataset based on simulation. In *2021 IEEE Int. Conf. on Robotics and Automation, ICRA*, 2020. 2, 3

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 2

[10] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 5, 6

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 2

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 3

[13] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations*, 2014. 1

[14] Baorui Ma, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Neural-pull: Learning signed distance function from point clouds by learning to pull space onto surface. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7246–7257. PMLR, 18–24 Jul 2021. 2

[15] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 5

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 2, 3

[17] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*, pages 523–540. Springer, 2020. 1, 2

[18] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1, 3

[19] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 1

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. 1

[21] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020. 5

[22] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019. 2

[23] Phil Wang. Dalle2-pytorch, 2022. 1

[24] Rundi Wu, Xuelin Chen, Yixin Zhuang, and Baoquan Chen. Multimodal shape completion via conditional generative adversarial networks. In *European Conference on Computer Vision*, pages 281–296. Springer, 2020. 3

[25] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4541–4550, 2019. 3

[26] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In *ICCV*, 2021. 2, 4