Appendix of *Controllable Mesh Generation Through Sparse Latent Point Diffusion Models*.

Contents

A. Method And Network Architectures

A.1 Details of Shape as Point (SAP)

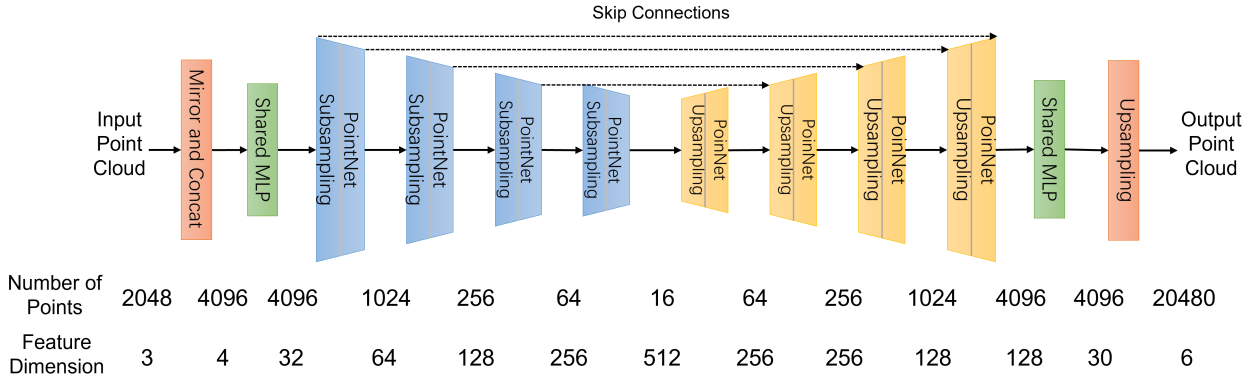

Figure 1: The architecture of the upsampling network that we use in SAP.

In this work, we choose the learning-based method proposed in [6] to reconstruct meshes from point clouds, we refer it as Shape as Points (SAP). SAP is composed of an upsampling network and a Differentiable Poisson Surface Reconstruction (DPSR) algorithm. The upsampling network upsamples the point cloud and estimates normals of every point in the upsampled point cloud. The DPSR algorithm solves a Poisson equation defined by the dense point cloud with normals using the spectral method and obtains an indicator field discretized on a 3D grid. Then the mesh is recovered by extracting the zero-isosurface of the 3D indicator grid. We refer readers to the original work [6] for details of the DPSR algorithm.

For the upsampling network, we use the improved PointNet++ proposed in [5]. The improved PointNet++ is able to extract features for every point in a point cloud. The features contain both global information about the overall 3D shape and local details. To upsample the input point cloud by a factor of γ, we can interpret the first part of the extracted feature for each point as γ spatial displacements. We can add the γ displacements to the original position of every input point to obtain γ new points. We interpret the second part of the extracted feature as γ normals. The method to obtain normals of the upsampled point cloud depends on whether the input point cloud has normals. Remind that our method is able to generate point clouds with normals, while baselines can only generate point clouds without normals. If the input point cloud has normals, we add the predicted γ normals for each point to the original normal and obtain γ refined normals for the upsampled points. If the input point cloud does not have normals, the predicted γ normals for each point are directly regarded as the normals of the upsampled points. In this way, we can upsample the input point cloud by a factor of γ and obtain normals for every upsampled point. We train two separate SAP models to handle the two cases where the input point cloud has normals and no normals, and refer them as SAP-Normal and SAP-No-Normal, respectively. When visualizing meshes reconstructed from point clouds generated by our method, we use SAP-Normal, but when computing the quantitative metrics, 1-NN, COV, and MMD for the meshes, we use SAP-No-Normal for both our method and baselines for a fair comparison.

We observe that most shapes in ShapeNet have symmetry about the xy plane. To encourage the reconstructed mesh to be symmetric about the xy plane, we mirror the input point cloud about the xy plane and concatenate the mirrored point cloud with the original point cloud, and then feed the concatenated point cloud to the upsampling network. To handle the case where the generated shape is not symmetric about the xy plane, we assign a label of 1 to the original points and a label of -1 to the mirrored points, so that the upsampling network can distinguish the original point cloud and mirrored point cloud.

We assume the number of input points to the upsampling network is 2048. We first mirror the input point cloud and obtain the concatenated point cloud that contains 4096 points. The upsampling network upsamples the 4096 points by a factor of $\gamma = 5$ and obtains a dense point cloud that contains 20480 points with normals. The detailed architecture of the upsampling network is shown in Figure 1. We adopt the same training procedures as [6] to train the upsampling network. We refer readers to [6] for details of the training algorithm. We train the network on all the 13 categories of the ShapeNet dataset pre-processed by [6]. It is trained for 1000 epochs with a batchsize of 32 using the Adam optimizer with a learning rate of 2×10^{-4}. The checkpoint with the lowest loss is used to reconstruct meshes from

point clouds generated by our method and baselines.

We find that the inputs to the SAP module during training and testing have discrepancies: The inputs during training are point clouds sampled from mesh surfaces, while the inputs during testing are point clouds generated by our method or baselines, possibly with flaws and are not seen by the SAP module during training. We tackle this problem by adding some Gaussian noises with a standard deviation of 0.02 to the input point clouds during training, and add Gaussian noises of the same magnitude to input point clouds during testing, in the hope that the Gaussian noises will submerge the flaws in the generated point clouds.

We conduct an ablation study on the effect of utilizing normals, symmetry, and adding Gaussian noises in Section B.9.

A.2 Details of the Feature Transfer (FT) Module

As mentioned in Section 3.1 in the main text, we use the Feature Transfer (FT) module in our autoencoder. The FT module is originally proposed in [5]. We briefly repeat the design of the FT module. The FT module can map features from one set of points to the second set of points. Assume the first set of points is $\boldsymbol{X}^1 = \{x_j^1 \in \mathbb{R}^3 | 1 \leq j \leq N_1\}$ with features $\boldsymbol{F}^1 = \{f_j^1 \in \mathbb{R}^{d^1} | 1 \leq j \leq N^1\}$, and the second set of points is $\boldsymbol{X}^2 = \{x_j^2 \in \mathbb{R}^3 | 1 \leq j \leq N_2\}$ with features $\boldsymbol{F}^2 = \{f_j^2 \in \mathbb{R}^{d^2} | 1 \leq j \leq N^2\}$. For each point x_j^2 in $\boldsymbol{X}^2$, we find its K-nearest neighbors in $\boldsymbol{X}^1$. Features at the neighbors are transformed through a shared Multi-layer Perceptron (MLP), then aggregated to the point x_j^2 through the self-attention mechanism proposed in [5]. Note that the original feature f_j^2 at x_j^2 are used as queries in the self-attention mechanism. We refer readers to [5] for details of the self-attention mechanism. In this way, we can map features $\boldsymbol{F}^1$ at $\boldsymbol{X}^1$ to features at $\boldsymbol{X}^2$.

A.3 Network Architectures

Architecture of the latent DDPMs. As mentioned in Section 3.2 in the main text, we train two DDPMs in the latent space of our point cloud autoencoder. The first one learns the distribution of the sparse latent points, and its architecture is shown in Figure 2. The second DDPM learns the distribution of the features at the sparse latent points, and its architecture is shown in Figure 3.

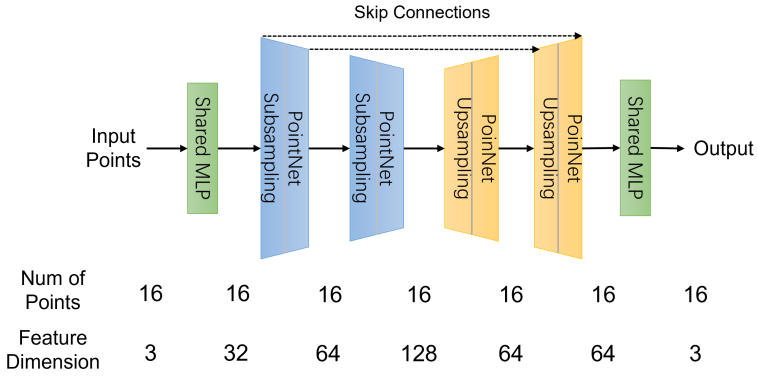

Figure 2: The architecture of the latent DDPM that learns the distribution of the sparse latent points.

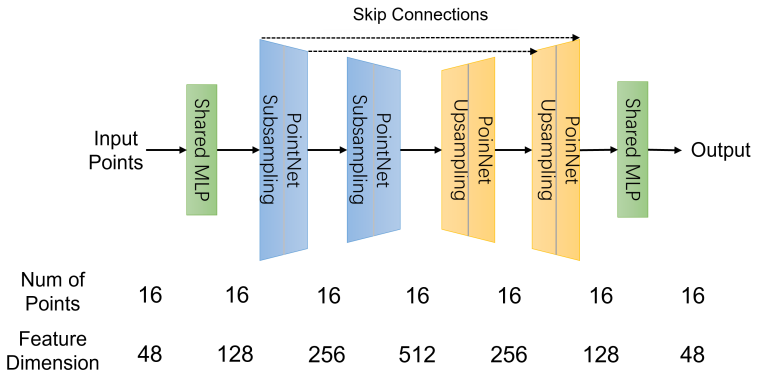

Figure 3: The architecture of the latent DDPM that learns the distribution of features at the sparse latent points.

A.4 DDPM Hyperparameters

In Section 2.1 in the main text, we define constants $\beta_t, t = 1, 2, \ldots, T$ in the DDPM. In all our experiments, we use a linear schedule for β_t's. β_1 is set to 1×10^{-4}, β_T is set to 2×10^{-2}, and T is set to 1000.

A.5 Sampling Algorithm

In Section 5.2 in the main text, we mention that we can sample features only for a portion of the sparse latent points, while fixing the features of rest points, so that we can only change the shape of a part of an object and keep other parts fixed. We describe the detailed sampling method in the following algorithm.

Algorithm 1 Sample new features for a part of sparse latent points.

Input: The second DDPM $\boldsymbol{\epsilon}_\phi$. Positions of the sparse latent points $\boldsymbol{X} \in \mathbb{R}^{N\times 3}$, features at every point $\boldsymbol{F} \in \mathbb{R}^{N\times d}$, where N is the number of the sparse latent points and d is the dimension of the features. A mask $\boldsymbol{m} \in \mathbb{R}^N$ that contains 0 and 1, where 1 denotes points that we want to sample new features for and 0 denotes points that we keep the original features.

Output: The sampled features $\boldsymbol{F}^0 \in \mathbb{R}^{N\times d}$.

Sample $\boldsymbol{f}^T$ of shape $(N \cdot d, 1)$ from standard Gaussian distribution
$\boldsymbol{x} \leftarrow$ reshape $\boldsymbol{X}$ to shape $(N \cdot 3, 1)$
$\boldsymbol{M} \leftarrow$ repeat $\boldsymbol{m}$ for d times and concatenate them to a matrix of shape (N, d)
for $t = T, T-1, \ldots, 1$ **do**
    $\tilde{\boldsymbol{f}}^0 = \frac{\boldsymbol{f}^t}{\sqrt{\bar{\alpha}_t}} - \frac{\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\phi(\boldsymbol{f}^t, \boldsymbol{x}, t)}{\sqrt{\bar{\alpha}_t}}$
    $\tilde{\boldsymbol{F}}^0 \leftarrow$ reshape $\tilde{\boldsymbol{f}}^0$ to shape (N, d)
    $\tilde{\boldsymbol{F}}^0 \leftarrow \boldsymbol{M} \odot \tilde{\boldsymbol{F}}^0 + (\boldsymbol{1} - \boldsymbol{M}) \odot \boldsymbol{F}$, where $\odot$ is element-wise product.
    $\tilde{\boldsymbol{f}}^0 \leftarrow$ reshape $\tilde{\boldsymbol{F}}^0$ to shape $(N \cdot d, 1)$
    Sample $\boldsymbol{f}^{t-1}$ from $q(\boldsymbol{f}^{t-1} \mid \boldsymbol{f}^t, \tilde{\boldsymbol{f}}^0) = \mathcal{N}(\boldsymbol{f}^{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\tilde{\boldsymbol{f}}^0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{f}^t, \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \mathbf{I})$
end for
$\boldsymbol{F}^0 \leftarrow$ reshape $\boldsymbol{f}^0$ to shape (N, d)
return $\boldsymbol{F}^0$

B. Experiments

B.1 Dataset Details

We use ShapeNet [2] to train our mesh generative model and compare it with other baselines. We use the pre-processed ShapeNet dataset provided by [6]. It contains 13 categories of objects: airplane, bench, cabinet, car, chair, display, lamp, loudspeaker, rifle, sofa, table, telephone, and watercraft. It splits the dataset into a training set and a validation set. For each shape, it provides 10000 points with normals sampled from the mesh surface, and the signed distance field (SDF) of the shape discretized on a 128^3 grid. Most shapes in it are normalized in a unit bounding box, namely, the center of the object's bounding box is placed at the origin, and the maximum length of the three sides of the bounding box is scaled to 1 such that each shape ranges from -0.5 to 0.5. We scale the shapes in the dataset by a factor of 2 to make them range from -1 to 1.

When evaluating point clouds generated by our method and baselines, or point clouds sampled from the reconstructed meshes, we normalize each point cloud according to its bounding box using the same method described above such that each point cloud ranges from -1 to 1.

B.2 Training Details

We first train autoencoders on 5 categories of ShapeNet: Airplane, cabinet, car, chair, and lamp. Each autoencoder is trained for 6000 epochs, and we save a checkpoint every 100 epoch. The checkpoint with the lowest reconstruction error is chosen for the next step latent DDPM training. The autoencoder is trained using the Adam optimizer with a learning rate of 0.001 and batchsize is 32. The autoencoder's reconstruction error (CD loss) on the validation set is $\{0.81, 3.94, 2.15, 2.86, 1.71\} \times 10^{-3}$ for airplane, cabinet, car, chair, and lamp, respectively. Note that the shapes in the dataset are scaled by a factor of 2 and therefore the CD loss is amplified as well.

After training the autoencoder, we fix its parameters and train two latent DDPMs in its latent space. The first DDPM learns the distribution of the positions of the sparse latent points. It is trained for 10000 epochs using the Adam optimizer with a learning rate of 0.0002 and batchsize is 32. We record the exponential moving average (EMA) of the DDPM parameters along the training trajectory. An EMA rate of 0.9999 is used for airplane and car, and an EMA rate of 0.999 is used for cabinet, chair, and lamp. The second DDPM learns the distribution of the features at the sparse latent points. It is trained for 10000 epochs using the Adam optimizer with a learning rate of 0.0002 and batchsize is 32. An EMA rate of 0.999 is used for all categories. For all baselines, we train them using their released codebase and follow the default setting.

B.3 Normal Consistency Loss

For evaluation in Section 5.2 in the main text, we use the normal consistency loss between two point clouds $\boldsymbol{X}$ and $\boldsymbol{Y}$:

$$L_{\text{normal}} = \sum_{\boldsymbol{x} \in \boldsymbol{X}} [1 - |\cos(\boldsymbol{n}_x, \boldsymbol{n}_{y^*})|] + \sum_{\boldsymbol{y} \in \boldsymbol{Y}} [1 - |\cos(\boldsymbol{n}_{x^*}, \boldsymbol{n}_y)|],$$

where $\boldsymbol{n}_x, \boldsymbol{n}_y$ denotes the normal of the points $\boldsymbol{x}, \boldsymbol{y}$, and $\boldsymbol{y}^* = \text{argmin}_{\boldsymbol{y} \in \boldsymbol{Y}} ||\boldsymbol{x} - \boldsymbol{y}||$, $\boldsymbol{x}^* = \text{argmin}_{\boldsymbol{x} \in \boldsymbol{X}} ||\boldsymbol{x} - \boldsymbol{y}||$.

For training of the autoencoder described in Section 3.1 in the main text, we use a modified normal consistency loss:

$$L'_{\text{normal}} = \sum_{\boldsymbol{x} \in \boldsymbol{X}} ||\boldsymbol{n}_x - \boldsymbol{n}_{y^*}||^2 + \sum_{\boldsymbol{y} \in \boldsymbol{Y}} ||\boldsymbol{n}_{x^*} - \boldsymbol{n}_y||^2.$$

B.4 Complete Evaluation Results for Generated Meshes.

In this section, we demonstrate the complete evaluation results for meshes generated by our method and baselines. We train two variants of our method. They are different in how to choose the initial point in FPS when sampling the sparse latent points. The first one chooses the centroid of the input point cloud as the initial point, while the second one randomly chooses a point in the input point cloud as the initial point. The evaluation metrics we use are 1-NN, MMD, and COV. These metrics may not accurately reflect the real quality of generated meshes. See qualitative results in Figure 4 for visual comparison. We find that MMD with normal consistency loss best matches human perception, possibly because MMD with normal consistency loss is more sensitive to the discrepancies of the surface curvatures between the generated meshes and reference meshes.

Table 1: 1NN-Acc (Percentage) comparison of meshes generated by our method and baselines. "N.C." denotes normal consistency loss. "Centroid" denotes the FPS method in which we choose the centroid of the point cloud as the initial point, and "Random" denotes the FPS method in which we randomly choose a point as the initial point.

	Airplane			Cabinet			Car			Chair			Lamp		
	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.
TreeGan [7]	81.31	71.78	84.65	69.75	74.20	55.73	91.32	75.23	87.18	70.24	62.04	56.94	74.68	59.74	**50.43**
ShapeGF [1]	73.39	71.29	76.36	61.46	**53.82**	56.37	**67.29**	**59.01**	54.47	56.57	54.14	54.65	57.79	**51.30**	54.33
PVD [8]	88.86	83.91	86.14	60.82	65.28	58.60	70.09	59.87	58.14	**55.39**	52.36	55.10	58.44	56.06	76.19
DPM [4]	75.50	68.81	**50.37**	62.42	63.69	**49.68**	86.38	79.44	**50.00**	66.17	69.50	**49.93**	66.23	61.04	52.60
SPGAN [3]	82.05	70.42	83.29	70.38	61.15	60.19	85.18	75.23	59.61	79.03	75.85	77.99	67.97	64.07	70.13
Ours (Centroid)	**70.17**	**65.84**	72.40	55.73	59.24	55.41	69.09	64.62	53.40	56.72	**51.18**	53.77	58.44	58.87	52.81
Ours (Random)	72.40	67.82	72.52	**53.50**	56.69	54.78	70.09	63.68	53.34	56.57	53.10	53.91	**56.28**	54.11	54.33

Table 2: MMD comparison of meshes generated by our method and baselines. CD, EMD, and normal consistency (N.C.) losses are multiplied by 1000, 100, and 10, respectively. We find that the N.C. loss computed MMDs best match human evaluations based on examples in Figure 4.

	Airplane			Cabinet			Car			Chair			Lamp		
	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.
TreeGan [7]	5.01	4.04	3.79	10.51	7.42	4.04	5.17	3.59	3.94	16.10	9.02	5.06	26.68	12.28	6.31
ShapeGF [1]	**4.33**	4.03	3.26	9.75	**6.49**	2.98	**4.46**	**3.10**	3.40	14.43	8.53	4.53	19.99	**11.04**	5.43
PVD [8]	5.11	4.29	3.42	11.01	7.24	3.51	4.63	3.12	3.47	15.89	**8.38**	4.31	24.85	12.80	5.01
DPM [4]	4.46	4.00	4.80	9.67	7.39	4.69	4.99	3.57	4.95	14.09	9.51	6.20	**19.63**	12.02	6.84
SPGAN [3]	5.09	4.03	3.39	10.52	7.18	3.04	5.02	3.49	3.54	18.86	9.67	4.68	23.64	12.63	4.87
Ours (Centroid)	4.46	**3.77**	**3.07**	9.52	6.50	**2.76**	4.55	3.20	**3.23**	14.76	8.49	4.19	23.61	11.87	4.38
Ours (Random)	4.37	3.81	**3.07**	**9.37**	6.77	2.79	4.58	3.18	**3.23**	14.87	8.63	**4.18**	21.48	11.37	**4.32**

Table 3: COV (Percentage) comparison of meshes generated by our method and baselines. "N.C." denotes normal consistency loss.

	Airplane			Cabinet			Car			Chair			Lamp		
	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.
TreeGan [7]	46.29	43.81	31.68	43.31	**50.96**	29.94	33.24	34.18	8.28	49.63	48.89	26.29	47.19	46.75	29.87
ShapeGF [1]	**49.50**	41.58	**41.34**	45.86	49.04	38.85	**44.73**	**47.80**	14.82	**52.88**	**50.96**	34.12	50.65	**55.84**	**36.80**
PVD [8]	34.41	34.9	32.67	45.85	48.41	30.57	40.05	41.39	**30.68**	48.15	50.52	32.20	51.08	55.41	36.79
DPM [4]	43.32	**48.76**	32.18	49.68	50.32	33.76	33.24	35.11	7.21	44.02	47.27	26.44	48.48	52.38	29.44
SPGAN [3]	41.09	46.29	34.65	39.49	42.68	32.48	32.58	36.72	12.68	30.13	31.91	21.86	46.75	49.78	34.63
Ours (Centroid)	48.51	46.29	39.85	**52.87**	47.77	37.58	38.99	39.92	12.55	49.19	49.63	34.71	**52.81**	52.38	36.36
Ours (Random)	48.76	47.03	37.87	50.96	**50.96**	**40.76**	43.12	41.26	13.89	48.89	50.37	**34.86**	51.08	52.81	33.77

B.5 Meshes Generated by Our Method and Baselines.

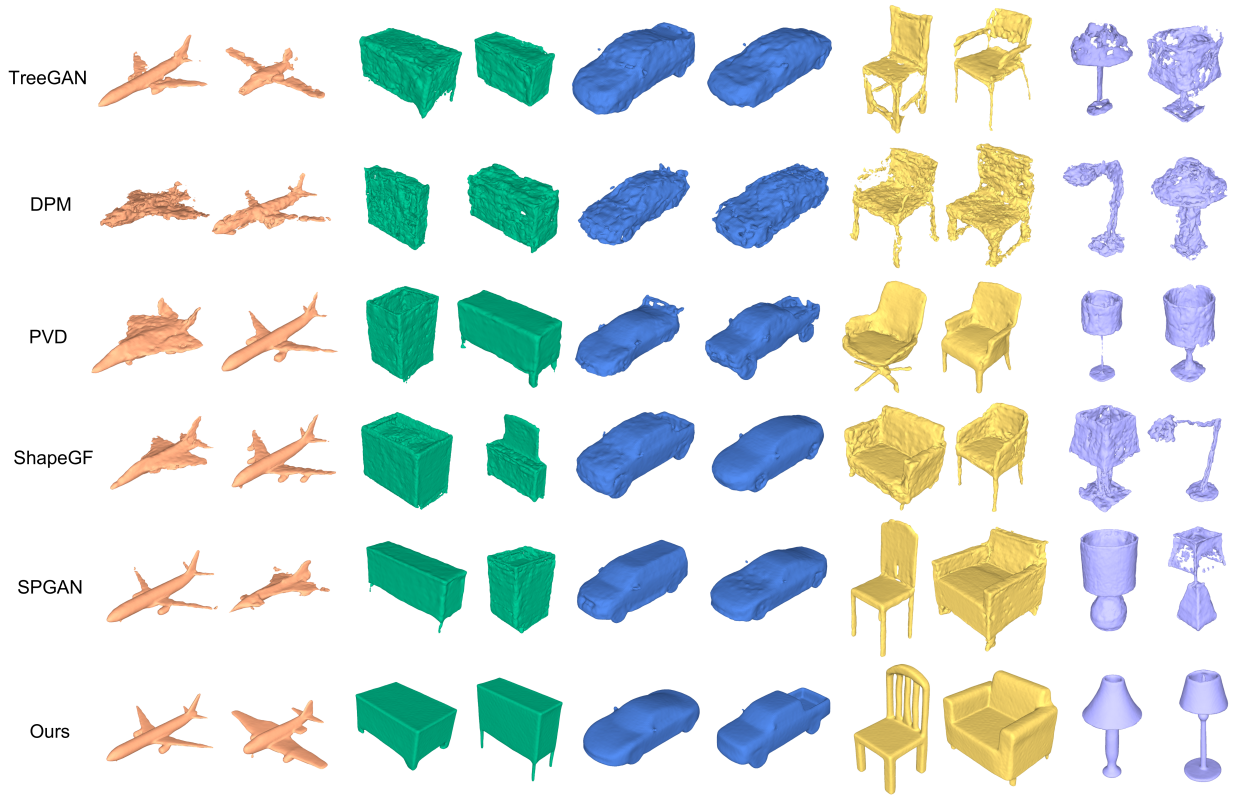

Figure 4: Compare meshes generated by our method and baselines. We can see that meshes generated by our method have smoother surfaces and sharper local details.

B.6 Complete Evaluation Results for Generated Point Clouds.

In this section, we demonstrate the complete evaluation results for point clouds generated by our method and baselines. We train two variants of our method. They are different in choosing the initial point in FPS when sampling the sparse latent points. The first one chooses the centroid of the input point cloud as the initial point, while the second one randomly chooses a point in the input point cloud as the initial point. The metrics we use are 1-NN, MMD, and COV. These metrics may not accurately reflect the real quality of generated point clouds. See qualitative results in Figure 5 for visual comparison.

Table 4: 1NN-Acc (Percentage) comparison of point clouds generated by our method and baselines. "Centroid" denotes the FPS method in which we choose the centroid of the point cloud as the initial point, and "Random" denotes the FPS method in which we randomly choose a point as the initial point.

	Airplane		Cabinet		Car		Chair		Lamp	
	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD
TreeGan [7]	79.95	99.26	71.66	99.68	91.12	97.40	73.86	94.83	69.70	93.72
ShapeGF [1]	64.23	74.01	59.87	62.74	63.08	64.49	**54.65**	69.35	56.06	56.49
PVD [8]	85.52	81.06	61.46	60.82	64.28	**53.81**	55.68	53.24	56.92	56.06
DPM [4]	77.48	**65.47**	68.47	65.29	82.18	69.43	63.15	61.30	66.67	64.29
SPGAN [3]	79.70	80.69	70.38	86.31	81.51	83.58	79.69	83.01	69.26	69.48
Ours (Centroid)	**62.50**	70.79	54.78	63.69	**58.21**	63.28	55.17	**52.66**	53.68	**55.19**
Ours (Random)	64.36	75.74	**51.27**	**58.28**	58.28	64.22	57.02	58.12	**53.25**	56.49

Table 5: MMD comparison of point clouds generated by our method and baselines. CD and EMD losses are multiplied by 1000 and 100, respectively.

	Airplane		Cabinet		Car		Chair		Lamp	
	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD
TreeGan [7]	4.24	9.54	9.74	23.10	4.86	4.74	15.28	12.83	21.06	19.75
ShapeGF [1]	3.96	4.03	9.35	6.84	4.28	3.20	13.83	9.39	17.92	**10.83**
PVD [8]	4.50	3.93	11.07	7.58	4.53	3.05	14.68	8.29	20.60	11.47
DPM [4]	**3.85**	**3.84**	9.23	7.22	4.67	3.22	**12.44**	8.80	**17.13**	12.07
SPGAN [3]	4.94	4.15	10.32	8.49	4.82	3.64	18.60	10.03	21.97	13.24
Ours (Centroid)	4.17	**3.84**	8.72	6.47	**4.05**	3.11	14.00	8.26	21.99	11.79
Ours (Random)	4.05	4.09	**8.57**	**6.46**	4.07	**2.99**	14.01	**8.40**	20.34	11.15

Table 6: COV (Percentage) comparison of point clouds generated by our method and baselines.

	Airplane		Cabinet		Car		Chair		Lamp	
	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD
TreeGan [7]	47.28	15.84	43.95	12.74	37.52	23.23	48.01	22.90	48.05	25.97
ShapeGF [1]	**52.23**	40.35	49.04	**52.87**	45.39	43.93	**51.85**	41.36	52.38	**55.84**
PVD [8]	35.15	40.59	52.86	47.77	43.25	**47.53**	47.41	**51.40**	52.38	51.51
DPM [4]	41.83	**49.01**	52.23	48.41	28.57	42.06	46.82	48.89	49.78	51.52
SPGAN [3]	39.36	33.66	42.68	36.94	32.18	37.25	28.95	25.55	46.75	48.05
Ours (Centroid)	48.76	42.08	52.87	47.13	40.19	40.05	49.93	49.93	**53.68**	52.38
Ours (Random)	49.50	37.62	**54.78**	49.68	**46.19**	41.66	49.19	50.52	50.65	53.68

B.7 Point Clouds Generated by Our Method and Baselines.

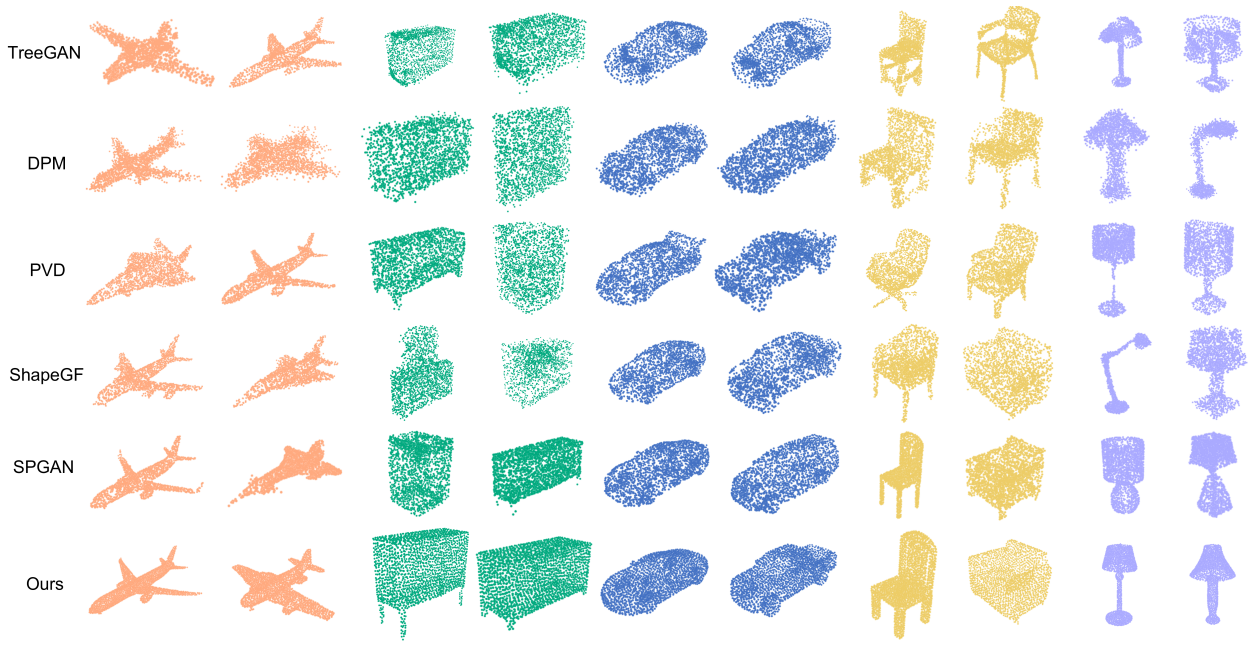

Figure 5: Compare point clouds generated by our method and baselines. We can see that point clouds generated by our method distribute more uniformly on the surface of the objects, and have sharper local details.

B.8 More examples generated by our method.

In this section, we show more shapes generated by our methods in Figure 6. We also train our method on two additional categories: Rifle and vessel.

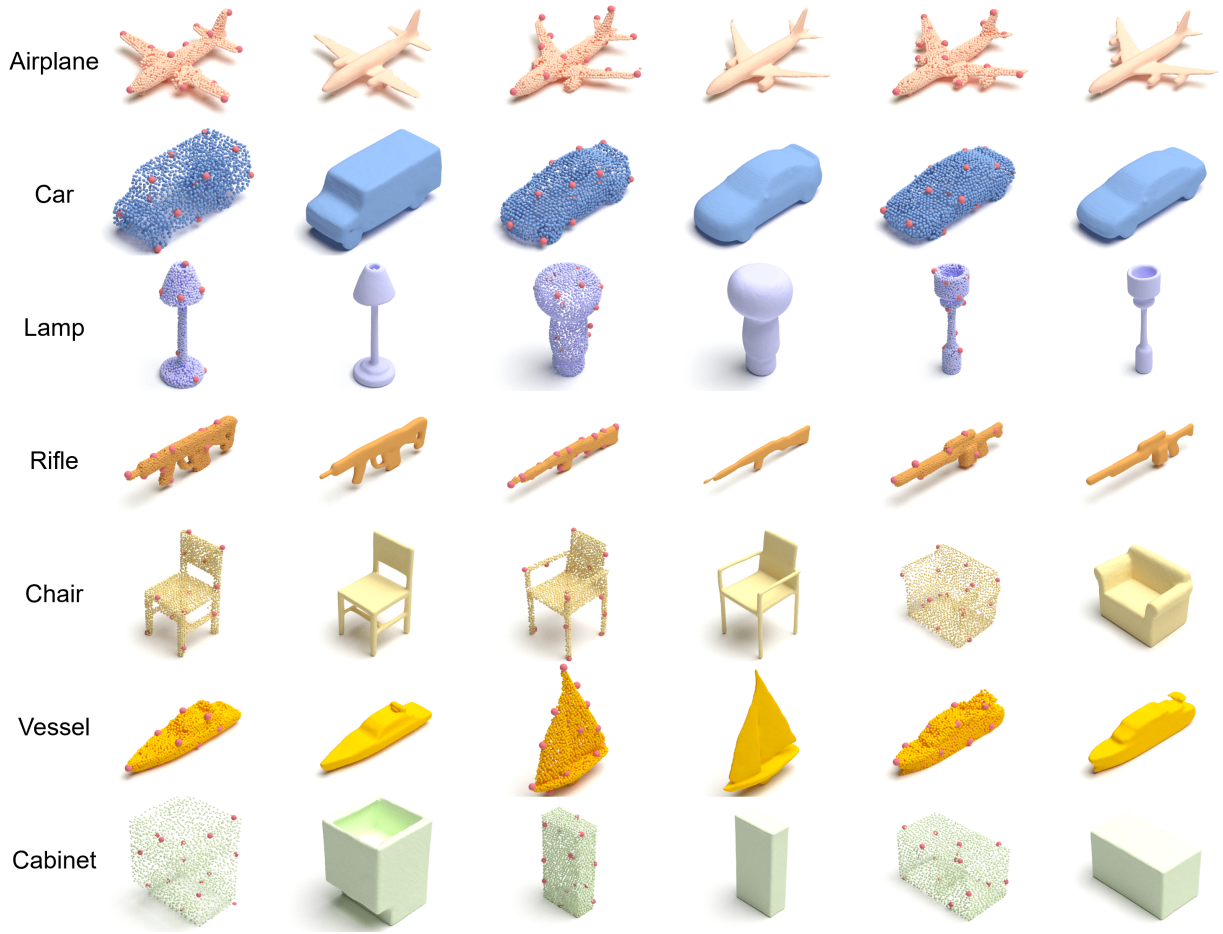

Figure 6: More point clouds and meshes generated by our method.

B.9 Ablation Study on the SAP Module.

As mentioned in Section A.1, for the SAP module, we can choose whether to utilize symmetry of the 3D shapes, and normals of the input point clouds, and whether to add Gaussian noises to the input point clouds during training and testing. In this section, we conduct an ablation study on the 3 techniques in the SAP module. Specifically, we use our method to generate point clouds on the five categories of ShapeNet (airplane, cabinet, car, chair, and lamp), and use SAP modules with different settings to reconstruct meshes from the generated point clouds. Then we compute metrics (1-NN, MMD, and COV) for meshes reconstructed by different SAP settings and compare their performance. Results are shown in Table 7, Table 8 and Table 9.

We can see that utilizing symmetry brings improvements to these metrics in most cases, while normals do not have an obvious impact, and Gaussian noises usually lead to a drop in these metrics. These metrics may not correlate well with human perceptions. Therefore, we also conduct a visual comparison of different SAP settings in Figure 7. We can see that utilizing symmetry can improve the symmetry of reconstructed meshes in many cases, utilizing normals can reconstruct meshes with smoother surfaces, and adding noises can help obtain smoother surfaces and fix flaws in the input point cloud.

Considering the quantitative and qualitative comparisons of different SAP settings, we use SAP with symmetry when computing quantitative metrics such as 1-NN, MMD, and COV for our method and baselines, and use SAP with symmetry, normal, and noise when visualizing reconstructed meshes.

Table 7: 1NN-Acc (Percentage) comparison of meshes reconstructed with different SAP settings. "N.C." denotes normal consistency loss.

	Airplane			Cabinet			Car			Chair			Lamp		
	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.
Symmetry, Normal, Noise	75.37	69.06	75.87	59.87	58.60	56.05	73.36	71.50	**52.20**	57.68	54.06	**52.66**	59.09	57.79	54.11
Symmetry, Normal	71.29	**64.98**	73.02	57.32	60.83	56.37	70.23	65.69	52.94	57.02	52.36	52.73	**57.58**	56.06	54.11
Symmetry	**69.80**	65.84	**72.15**	**56.69**	**56.37**	55.73	**68.76**	**63.15**	53.47	**56.20**	**51.18**	54.14	58.66	56.49	53.46
Vannila	71.41	65.22	73.51	57.96	56.69	**54.78**	69.69	65.02	53.81	56.72	52.88	53.18	59.31	**54.98**	**52.81**

Table 8: MMD comparison of meshes reconstructed with different SAP settings. CD, EMD, and normal consistency (N.C.) losses are multiplied by 1000, 100, and 10, respectively.

	Airplane			Cabinet			Car			Chair			Lamp		
	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.
Symmetry, Normal, Noise	4.56	3.82	3.15	9.69	6.62	**2.71**	4.60	3.33	**3.15**	14.91	8.49	**4.14**	24.60	12.01	**4.32**
Symmetry, Normal	4.47	**3.70**	**3.08**	9.53	6.73	2.74	4.57	3.24	3.21	14.87	8.52	4.17	23.86	**11.74**	4.33
Symmetry	**4.45**	3.72	3.09	9.48	6.56	2.74	**4.55**	**3.17**	3.24	**14.74**	**8.43**	4.20	**23.56**	11.76	4.38
Vannila	4.48	3.76	3.11	**9.45**	**6.48**	2.75	4.57	3.18	3.24	14.77	**8.43**	4.19	23.65	11.93	4.39

Table 9: COV (Percentage) comparison of meshes reconstructed with different SAP settings. "N.C." denotes normal consistency loss.

	Airplane			Cabinet			Car			Chair			Lamp		
	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.
Symmetry, Normal, Noise	45.30	43.56	37.87	**52.87**	49.04	**39.49**	39.52	37.38	12.28	48.01	51.85	34.86	50.65	51.95	35.06
Symmetry, Normal	49.01	44.55	37.13	51.59	45.86	38.85	39.52	**40.99**	12.55	49.04	52.44	**36.19**	52.81	51.95	35.93
Symmetry	**50.00**	45.54	**38.86**	52.23	**50.96**	38.22	**39.65**	38.85	**12.95**	**49.34**	**54.21**	35.30	**54.11**	53.25	**36.36**
Vannila	49.50	**47.52**	37.87	52.23	50.32	**39.49**	39.39	40.45	12.28	48.89	52.73	34.42	52.81	**54.11**	**36.36**

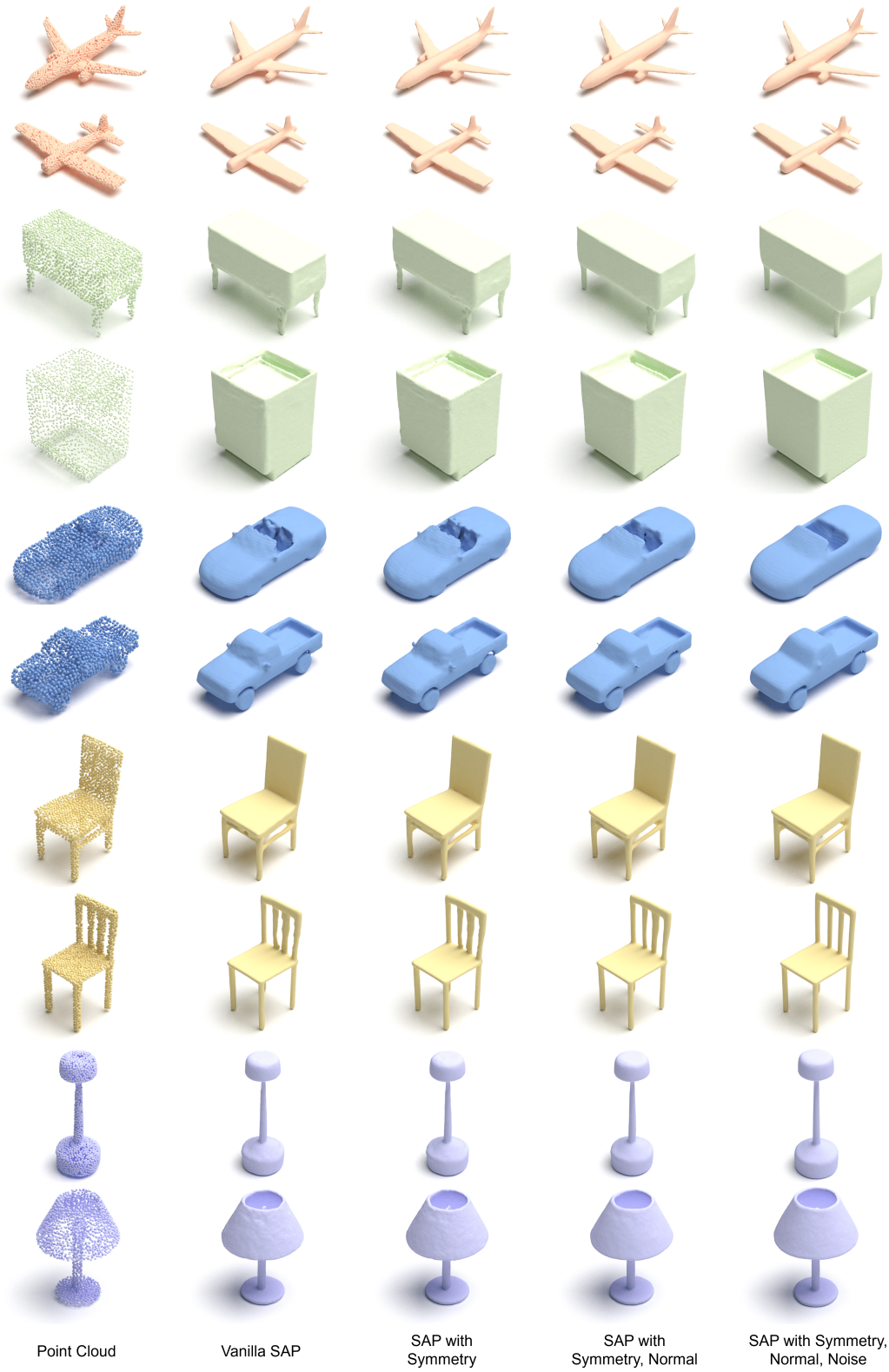

Figure 7: Qualitative comparisons of different SAP settings. The left column is point clouds generated by our method, and the right 4 columns are meshes reconstructed by SAP under different settings.

B.10 Ablation Study on the Sparse Latent Points

In this section, we conduct an ablation study on the number of sparse latent points (8,16,32) and the method to obtain them. The first method is to use FPS to obtain the sparse latent points from the input point cloud, and the initial point is chosen randomly. The second is to randomly sample points from the input point cloud as the sparse latent points. We train our autoencoders and latent DDPMs under different settings on three categories: Airplane, car, and chair. We report the autoencoder's reconstruction error and the latent DDPMs' average generation time of a single point cloud in Table 10. We also report 1-NN, MMD, and COV for point clouds and meshes generated under different settings in Table 11, Table 12, and Table 13, respectively.

We can see that increasing the number of latent points reduces the reconstruction error, but slows down the generation speed of our model. As for the generation quality reflected by the quantitative metrics in Table 11, Table 12, and Table 13, we find that the number of sparse latent points and the method to obtain them do not have a clear or significant impact. We therefore further add a visual comparison of these settings in Figure 8. We can see that these settings indeed generate point clouds and meshes of similar quality. The number of sparse latent points mainly affects the ability to perform controllable generation of our method. When we have too few sparse latent points, it restricts the complexity of the controls we can perform on an object. When we have too many sparse latent points, we can perform more delicate controls of the generated shapes, but it also becomes more complex and less intuitive to manipulate the sparse latent points since we need to consider their relative positions and ensure that they form a plausible skeleton of an object. Therefore, we choose to use 16 sparse latent points in our main experiments because it strikes a balance between the complexity of the controls we can perform and ease of use. As for the method to obtain the sparse latent points, although randomly choosing the latent points can generate point clouds and meshes of similar quality to those generated by using FPS to obtain the latent points, Figure 8 shows that latent points obtained by random sampling usually can not uniformly stretch over the surface of an object, while latent points obtained by FPS usually locate at semantically meaningful regions of an object and thus is more intuitive for controllable generation.

Table 10: Ablation study on the number of latent points (8,16,32) and the method to sample them (FPS or random sampling). We report the reconstruction error (CD $\times 10^{-3}$) of the autoencoder and the average generation time (seconds) per sample.

Number of Latent Points	Sampling Method	Airplane		Car		Lamp	
		Error	Time	Error	Time	Error	Time
8	FPS(Random)	0.94	**0.137**	2.22	**0.194**	2.18	**0.141**
16	FPS(Random)	0.81	0.197	2.15	0.242	1.70	0.196
32	FPS(Random)	**0.74**	0.414	**2.10**	0.450	**1.47**	0.414
16	Random	0.86	0.197	2.12	0.235	1.75	0.204

Table 11: Ablation study on the number of latent points (8,16,32) and the method to sample them (FPS or random sampling). We report 1NN-Acc (Percentage) of meshes and point clouds generated by our method under different settings. "N.C." denotes normal consistency loss.

Data Format	Number of Latent Points	Sampling Method	Airplane			Car			Lamp		
			CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.
Mesh	8	FPS(Random)	71.91	63.74	72.03	**68.62**	67.42	**53.00**	64.72	55.63	58.01
	16	FPS(Random)	**70.17**	67.08	**71.66**	70.09	**63.68**	53.34	**56.93**	53.90	**53.25**
	32	FPS(Random)	72.28	**62.13**	73.51	71.76	66.15	53.40	62.99	54.98	56.06
	16	Random	71.04	64.85	73.51	68.76	64.29	**53.00**	60.61	**52.81**	56.49
Point Cloud	8	FPS(Random)	61.14	**63.37**	–	57.21	67.49	–	56.06	58.23	–
	16	FPS(Random)	64.36	75.74	–	58.28	**64.22**	–	**53.25**	56.49	–
	32	FPS(Random)	68.07	66.96	–	60.08	66.15	–	54.76	53.25	–
	16	Random	**57.30**	65.72	–	**56.74**	65.89	–	54.33	**52.60**	–

Table 12: Ablation study on the number of latent points (8,16,32) and the method to sample them (FPS or random sampling). We report MMD of meshes and point clouds generated by our method under different settings. CD, EMD, and normal consistency (N.C.) losses are multiplied by 1000, 100, and 10, respectively.

Data Format	Number of Latent Points	Sampling Method	Airplane			Car			Lamp		
			CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.
Mesh	8	FPS(Random)	4.46	3.76	**3.02**	**4.50**	3.23	**3.21**	**22.48**	**11.29**	4.34
	16	FPS(Random)	**4.41**	3.90	3.06	4.58	3.18	3.23	22.59	11.31	**4.31**
	32	FPS(Random)	4.50	**3.75**	3.14	4.64	**3.14**	3.25	23.07	11.49	4.44
	16	Random	4.49	3.76	3.08	4.52	3.19	**3.21**	23.63	11.68	4.37
Point Cloud	8	FPS(Random)	4.14	**3.63**	–	**3.95**	3.13	–	21.55	11.24	–
	16	FPS(Random)	4.05	4.09	–	4.07	**2.99**	–	20.34	**11.15**	–
	32	FPS(Random)	4.30	3.80	–	4.15	3.14	–	21.70	11.55	–
	16	Random	**4.01**	3.70	–	4.04	3.08	–	**19.61**	11.16	–

Table 13: Ablation study on the number of latent points (8,16,32) and the method to sample them (FPS or random sampling). We report COV (Percentage) of meshes and point clouds generated by our method under different settings. "N.C." denotes normal consistency loss.

Data Format	Number of Latent Points	Sampling Method	Airplane			Car			Lamp		
			CD	EMD	N.C.	CD	EMD	N.C.	CD	EMD	N.C.
Mesh	8	FPS(Random)	45.05	44.31	39.85	42.19	41.92	**15.35**	49.35	51.08	35.06
	16	FPS(Random)	45.54	42.08	37.62	43.12	41.26	13.89	50.22	**59.74**	38.53
	32	FPS(Random)	47.03	44.80	**40.10**	43.12	**43.12**	12.28	**52.38**	55.41	34.63
	16	Random	**47.52**	**47.03**	39.60	**43.26**	42.06	13.08	48.48	53.25	**38.96**
Point Cloud	8	FPS(Random)	46.53	**46.53**	–	43.66	40.05	–	51.52	**56.28**	–
	16	FPS(Random)	49.50	37.62	–	**46.19**	41.66	–	50.65	53.68	–
	32	FPS(Random)	49.26	42.82	–	45.13	**43.12**	–	53.25	55.41	–
	16	Random	**50.74**	45.05	–	43.79	41.39	–	**54.11**	53.25	–

B.11 Shape Interpolation and Extrapolation

As mentioned in Section 5.3 in the main text, we can perform interpolation between two objects in the latent space of our autoencoder. To achieve this, we can interpolate both the positions and features between the corresponding latent points. The key is to establish the correspondence between the two sets of sparse latent points of the two objects. For the two lamps shown in Figure 7 in the main text, one lamp is obtained by moving the positions of the latent points of the other lamp. The correspondence between their latent points is clear. However, for two generally unrelated objects, it is non-trivial to find the correspondence between their latent points, because the two sets of sparse latent points are sampled independently by farthest point sampling (FPS) from the two objects.

To tackle this issue, we propose to first sample sparse latent points $\boldsymbol{X}^1_{\text{latent}}$ for the first object $\boldsymbol{X}^1$, where both $\boldsymbol{X}^1_{\text{latent}}$ and $\boldsymbol{X}^1$ are set of points. If the object is in the form of a mesh, $\boldsymbol{X}^1$ can be obtained by sampling points from the surface of it. For the second object $\boldsymbol{X}^2$, we do not directly sample its sparse latent points $\boldsymbol{X}^2_{\text{latent}}$ from $\boldsymbol{X}^2$, instead, its latent points are obtained by finding the closest neighbor in $\boldsymbol{X}^2$ for each point in $\boldsymbol{X}^1_{\text{latent}}$. In this way, we can obtain $\boldsymbol{X}^2_{\text{latent}}$ and the correspondence with $\boldsymbol{X}^1_{\text{latent}}$ is established. Although $\boldsymbol{X}^2_{\text{latent}}$ obtained in this way may not be perfect in the sense that $\boldsymbol{X}^2_{\text{latent}}$ has a different distribution from points directly sampled from $\boldsymbol{X}^2$ using FPS, we find that our autoencoder can still well reconstruct $\boldsymbol{X}^2$ by encoding $\boldsymbol{X}^2$ to features at $\boldsymbol{X}^2_{\text{latent}}$. The first reason is that $\boldsymbol{X}^2_{\text{latent}}$ obtained in this way still generally stretches over the whole surface of $\boldsymbol{X}^2$. The second reason is that our autoencoder is robust to the positions of the sparse latent points. As mentioned in Section 3.1 in the main text, we add noises to the positions of the sparse latent points during the training of the autoencoder.

We provide several examples of shape interpolation between two unrelated objects in Figure 9. Beyond interpolation, it is also possible to perform extrapolation as shown in Figure 9.

Decompose feature and position. Our sparse latent point representation has great flexibility to perform interpolation. As shown in Figure 9 in the main text, we can perform both global and local interpolation. We will show that we can also decompose the features and the positions of the sparse latent points during interpolation, namely, we can

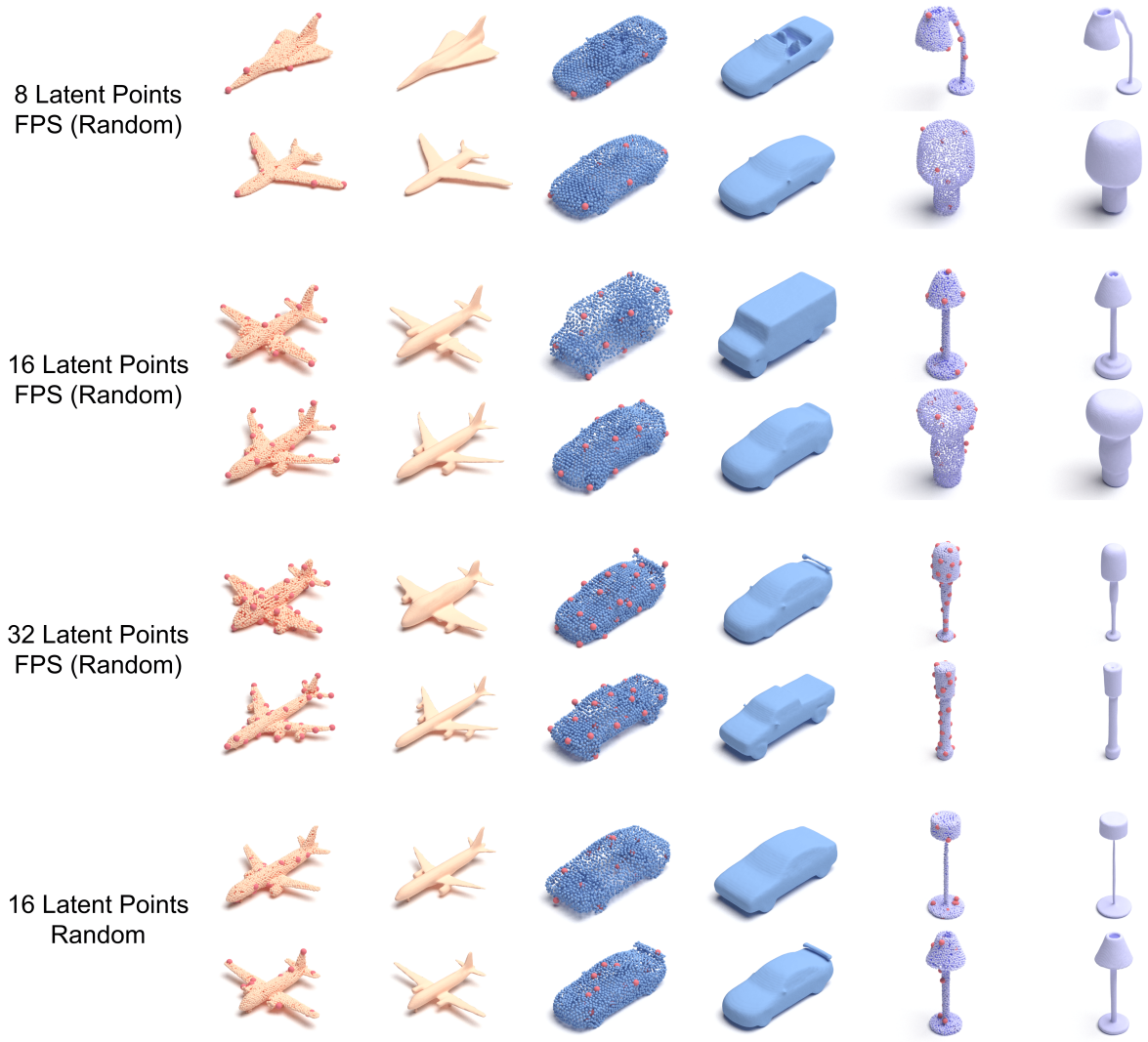

Figure 8: Ablation study on the number of latent points (8,16,32) and the method to sample them (FPS or random sampling). We show meshes and point clouds generated by our method under different settings.

only interpolate the features or the positions of the corresponding latent points during interpolation. We provide an example in Figure 10. We can see that the features of the latent points control the local geometry of the shape, while the positions of the latent points control the overall size of the shape.

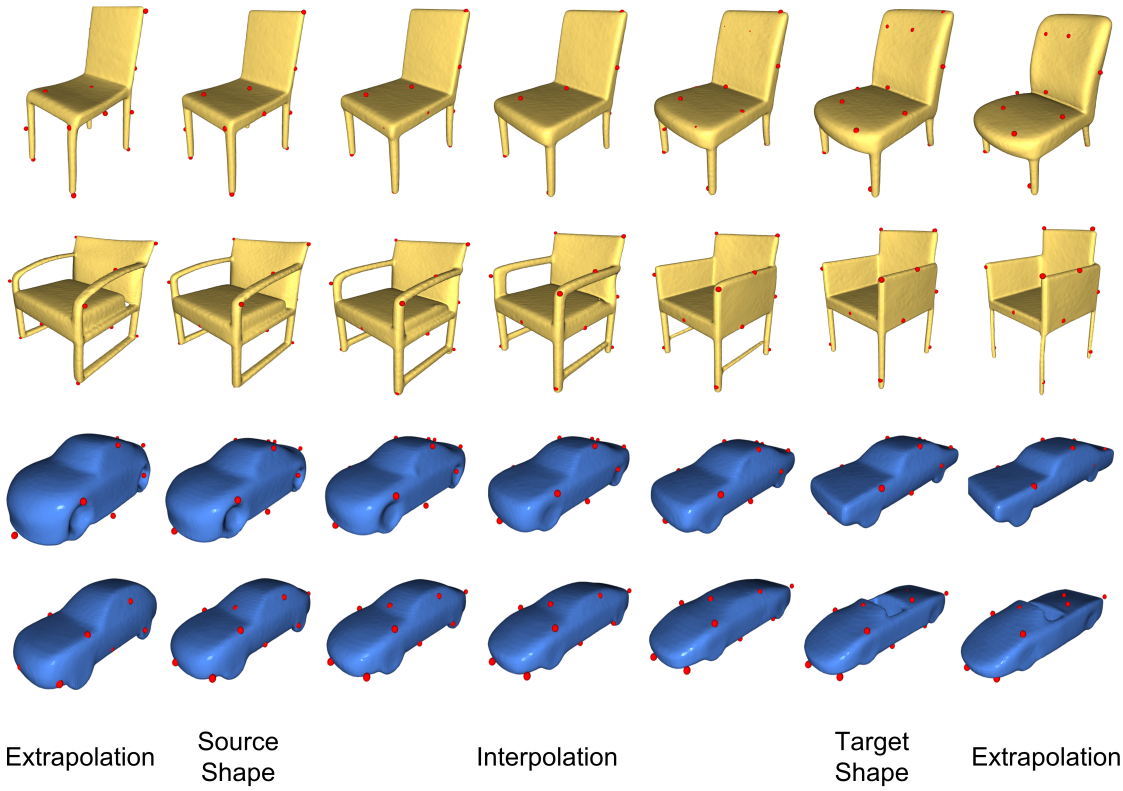

Figure 9: Interpolation and extrapolation between two unrelated objects. Note that some sparse latent points are within the mesh and are thus invisible.

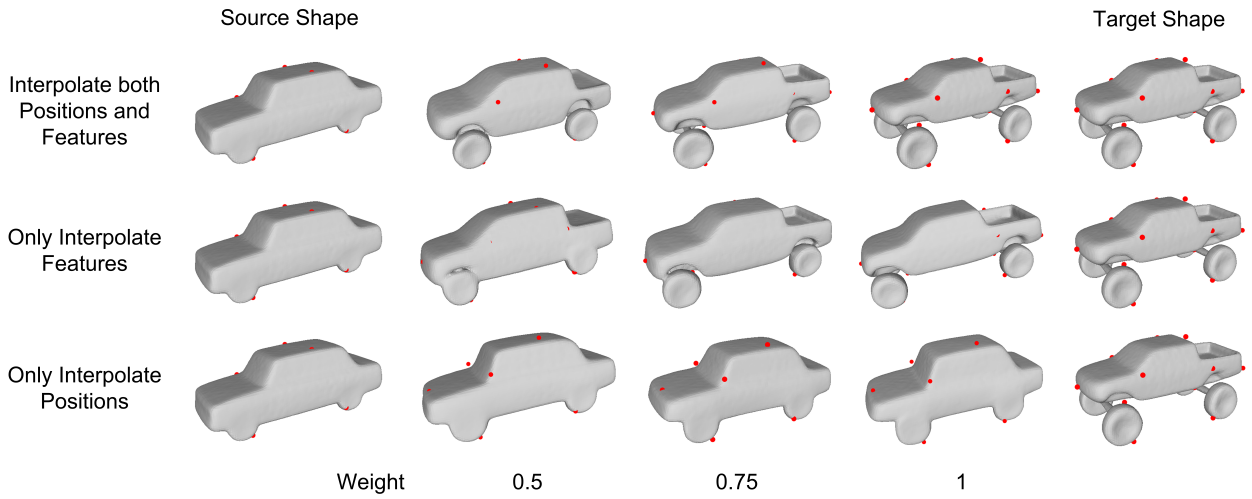

Figure 10: Our method can decompose the features and the positions of the sparse latent points during interpolation. In the first row, we interpolate both the positions and features of the sparse latent points. In the second row, we only interpolate features. In the third row, we only interpolate positions. Note that some sparse latent points are within the mesh and are thus invisible.

References

[1] R. Cai, G. Yang, H. Averbuch-Elor, Z. Hao, S. Belongie, N. Snavely, and B. Hariharan. Learning gradient fields for shape generation. In *European Conference on Computer Vision*, pages 364–381. Springer, 2020.

[2] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su,

et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[3] R. Li, X. Li, K.-H. Hui, and C.-W. Fu. Sp-gan: Sphere-guided 3d shape generation and manipulation. *ACM Transactions on Graphics (TOG)*, 40(4):1–12, 2021.

[4] S. Luo and W. Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2837–2845, 2021.

[5] Z. Lyu, Z. Kong, X. Xu, L. Pan, and D. Lin. A conditional point diffusion-refinement paradigm for 3d point cloud completion. *arXiv preprint arXiv:2112.03530*, 2021.

[6] S. Peng, C. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, and A. Geiger. Shape as points: A differentiable poisson solver. *Advances in Neural Information Processing Systems*, 34:13032–13044, 2021.

[7] D. W. Shu, S. W. Park, and J. Kwon. 3d point cloud generative adversarial network based on tree structured graph convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3859–3868, 2019.

[8] L. Zhou, Y. Du, and J. Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5826–5835, 2021.