StruMamba3D: Exploring Structural Mamba for Self-supervised Point Cloud Representation Learning

Supplementary Material

A. Overview

In this supplementary material, we first provide additional implementation details (Sec. B). Next, we present more experimental results (Sec. C) and qualitative analysis (Sec. D) to validate and analyze our proposed method. Finally, we discuss the limitations of our approach and potential directions for future research (Sec. E).

B. More Implementation Details

B.1. Structural SSM Block

In the state-wise update strategy and sequence-length adaptive strategy, we modify the SSM parameter generation process to enhance its efficiency. To provide a more intuitive understanding of the structural SSM block, we present its detailed operations in Algorithm 1. The process begins with the normalization of input points F_x and initial spatial states F_h followed by their linear projection into F_x^l and F_h^l. Next, a lightweight convolution is applied to F_x^l and F_h^l, yielding $\hat{F}_x$ and $\hat{F}_h$. Before processing $\hat{F}_x$ in both forward and backward directions, we compute the relative position offsets $\triangle P$ between the input points P_x and spatial states P_h. Based on the $\hat{F}_x$ and $\triangle P$, we generate the SSM parameters ($\mathbf{B}_o$, $\mathbf{C}_o$, $\mathbf{\Delta}_o'$). A learnable parameter τ is introduced to regulate the sampling interval $\mathbf{\Delta}_o'$, producing $\mathbf{\Delta}_o$. Utilizing $\mathbf{\Delta}_o$, we transform $\overline{\mathbf{A}}_o$, $\overline{\mathbf{B}}_o$. Finally, the SSM computes the output point features $\hat{F}_x^o$ and the updated spatial states $\hat{F}_h^o$. The forward and backward output point features, $\hat{F}_x^f$ and $\hat{F}_x^b$, are gated by F_z and then summed together. Similarly, the forward and backward spatial states, $\hat{F}_h^f$ and $\hat{F}_h^b$, are aggregated. The final outputs, F_x' and F_h', are obtained by connecting the updated inputs and spatial states to F_x and F_h using residual connections, respectively.

B.2. Pre-training Details

Following previous works [8], we pre-train our StruMamba3D on ShapeNet dataset [1], which contains 52,472 unique 3D models across 55 common object categories. For each shape, we sample 1024 points as input and partition them into 64 groups using FPS and KNN, with 60% of the groups being randomly masked. The pre-training process utilizes the AdamW [4] optimizer, incorporating cosine learning rate decay [7]. The initial learning rate is set to 1e-3, with a weight decay of 5e-2 and a dropout rate of 1e-1. The batch size is 128, and training is performed for 300 epochs. The model pre-training is conducted on a single NVIDIA RTX A6000 GPU and takes approximately 18 hours to complete.

B.3. Fine-tuning Details

Shape Classification on ScanObjectNN Dataset. ScanObjectNN is a challenging real-world dataset comprising approximately 15,000 objects across 15 distinct categories. For the shape classification task, we first sample 2048 points from each object as input. Next, we leverage our proposed StruMamba3D to extract discriminative group features, which are then aggregated through maximum and average pooling operations to obtain comprehensive global point cloud representations. These global features are processed through a classification head consisting of three linear layers, with respective output dimensions of 256, 256, and N_{cls} (the number of object categories). Finally, a cross-entropy loss is applied to supervise the predictions.

$$\mathcal{L}_{cls} = -\sum_{i=1}^{N_{cls}} y_i \log(p_i), \tag{1}$$

where y_i is the indicator function taking a value of 1 when the true class of the point cloud is i and 0 otherwise, and p_i denotes the predicted probability for class i. For the fine-tuning process, the model is trained for 300 epochs with a batch size of 32 and a learning rate of 5e-4. We utilize the AdamW optimizer [4] with a weight decay of 5e-2 and apply cosine learning rate decay [7], with an initial learning rate of 1e-6 and a warm-up period of 10 epochs.

Shape Classification on ModelNet40 Dataset. ModelNet40 is a widely used benchmark dataset for 3D shape classification, containing 12,311 unique 3D models spanning 40 categories. For the shape classification task, we begin by sampling 1,024 points from each object as input. All other experimental settings remain consistent with those used for the ScanObjectNN dataset. During testing, we report results both with and without the voting strategy. The voting strategy involves conducting ten tests with random scaling and averaging the predictions to obtain the final result.

Part Segmentation on ShapeNetPart Dataset. ShapeNetPart [1] is a benchmark dataset specifically designed for 3D part segmentation. It comprises 16,881 unique 3D models spanning 16 object categories and 50 part categories, providing a diverse and comprehensive dataset for evaluating fine-grained 3D shape understanding. For the part segmentation task, we begin by sampling 2,048 points from each shape as input. Following PointBERT [11], we adopt a similar segmentation head architecture and extract features from the 4-th, 8-th, and 12-th layers of our StruMamba3D. These three levels of features are concatenated and then pro-

Algorithm 1 Structural SSM Block Process

```
Require: Input point features $F_x$: (B, N, D), Initial spatial states $F_h$: (B, M, D)
Ensure: Output point features $F'_x$: (B, N, D), Final spatial states $F'_h$: (B, M, D)
 1: /* normalize the input point features $F_x$ */
 2: $F^n_x$: (B, N, D) $\leftarrow \text{Norm}(F_x)$, $F^n_h$: (B, M, D) $\leftarrow \text{Norm}(F_h)$
 3: $F^l_x$: (B, N, E) $\leftarrow \text{Linear}^x(F^n_x)$, $F^l_z$: (B, N, E) $\leftarrow \text{Linear}^z(F^n_x)$, $F^l_h$: (B, M, E) $\leftarrow \text{Linear}^h(F^n_h)$
 4: $\hat{F}_x$: (B, N, E) $\leftarrow \text{LightConv.}(F^l_x)$, $\hat{F}_h$: (B, M, E) $\leftarrow \text{LightConv.}(F^l_h)$
 5: /* process with different direction */
 6: for $o$ in$\{\text{forward}, \text{backward}\}$ do
 7:     /* $\triangle P$ is the relative offsets between the input points and spatial states: (B, N, M, 3) */
 8:     $\mathbf{B}_o$: (B, N, M) $\leftarrow \phi_{\mathbf{B}}(\hat{F}_x) + \text{MLP}_{\mathbf{B}}(\triangle P)$
 9:     $\mathbf{C}_o$: (B, N, M) $\leftarrow \phi_{\mathbf{C}}(\hat{F}_x) + \text{MLP}_{\mathbf{C}}(\triangle P)$
10:     /* softplus ensures positive $\mathbf{\Delta}_o$ */
11:     $\mathbf{\Delta}'_o$: (B, N, E) $\leftarrow \log(1 + \exp(\phi_{\mathbf{\Delta}}(\hat{F}_x)))$
12:     /* learnable parameter $\tau$ regulates the total sampling time $\mathbf{\Delta}_{all}$ */
13:     $\mathbf{\Delta}_o$: (B, N, E) $\leftarrow \tau \times \mathbf{\Delta}'_o / (\sum_{i=1}^{N} \mathbf{\Delta}'^{i}_o)$
14:     /* $\text{Parameter}^{\mathbf{A}}_o$ is learnable parameter: (M, E) */
15:     $\overline{\mathbf{A}}_o$: (B, N, M, E) $\leftarrow \mathbf{\Delta}_o \otimes \text{Parameter}^{\mathbf{A}}_o$
16:     $\overline{\mathbf{B}}_o$: (B, N, M, E) $\leftarrow \mathbf{\Delta}_o \otimes \mathbf{B}_o$
17:     $\hat{F}^o_x$: (B, N, E), $\hat{F}^o_h$: (B, M, E) $\leftarrow \mathbf{SSM}(\overline{\mathbf{A}}_o, \overline{\mathbf{B}}_o, \mathbf{C}_o)(\hat{F}_x, \hat{F}_h)$
18: end for
19: /* gated linear unit and residual connection */
20: $F'^f_x$: (B, N, E) $\leftarrow \hat{F}^f_x \odot \text{SiLU}(F_z)$, $F'^b_x$: (B, N, E) $\leftarrow \hat{F}^b_x \odot \text{SiLU}(F_z)$
21: $F'_x$: (B, N, C) $\leftarrow \text{Linear}^{out}(F'^f_x + F'^b_x) + F_x$
22: $F'_h$: (B, M, C) $\leftarrow \text{Linear}^h(\hat{F}^f_h + \hat{F}^b_h) + F_h$
23: return $F'_x$ and $F'_h$
```

cessed separately using average pooling and max pooling to obtain two distinct global features. Besides, leveraging feature propagation from PointNet++ [9], we upsample the concatenated features to match the 2,048 input points, ensuring per-point feature representation. The per-point features are then concatenated with the two global features and passed through a Multi-Layer Perceptron (MLP) for point-wise label prediction. To optimize the model, we use cross-entropy loss to supervise the predictions, which is formulated as follows:

$$\mathcal{L}_{seg} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N_{seg}} y^i_j \log(p^i_j), \tag{2}$$

where N represents the number of points, N_{seg} denotes the number of part categories, y^i_j is the indicator function taking a value of 1 when the true class of the point i is j and 0 otherwise, and p^i_j denotes the predicted probability for point i being classified as j. For fine-tuning, the model is trained for 300 epochs with a batch size of 32 and a learning rate of 2e-4. We utilize the AdamW optimizer [4] with a weight decay of 5e-2. Additionally, we apply cosine learning rate decay [7], starting with an initial learning rate of 1e-6 and incorporating a warm-up period of 10 epochs to stabilize early training.

Few-Shot Classification on ModelNet40 Dataset. In the n-way-k-shot setting, the support set consists of n distinct classes, each containing k samples. The model is trained only using the sampled $n \times k$ samples. For evaluation, we randomly sample 20 novel instances from each of the n classes to form the test set. We conduct few-shot classification experiments on the ModelNet40 dataset across four different configurations: 5-way-10-shot, 5-way-20-shot, 10-way-10-shot, and 10-way-20-shot. Each configuration is evaluated over 10 independent trials, and we report the mean accuracy along with the standard deviation. The training hyperparameters remain consistent with those used in the shape classification experiments on the ModelNet40 dataset.

C. More Experimental Results

C.1. Detailed Results on Part Segmentation

We present per-category part segmentation results on the ShapeNetPart [10] dataset. The results for PointGPT-S [2] and PointMamba [5] are reproduced using their official code. As shown in Tab. 1, our method achieves the best performance in most categories. The segmentation task requires the model to effectively capture the local structure of point clouds. Our approach enhances local structure modeling by introducing spatial states, which play a crucial role in improving segmentation performance. This spatial state modeling allows the network to better preserve spatial relationships and structural details, leading to superior results in segmentation tasks.

C.2. Large-scale 3D Scene Task

To evaluate the scalability of our approach to large-scale 3D scene understanding tasks, we conduct 3D object detection experiments on the ScanNetV2 dataset, which contains approximately 50K points per scene. For fair comparison, we follow the evaluation protocol of Point-M2AE and adopt 3DETR-m as our baseline. Our encoder is configured with the same number of layers as Point-M2AE, processing 2048 input tokens and employing 128 spatial states. We pretrain our model on ScanNetV2 for 1080 epochs using a learning rate of 5e-4 and a batch size of 16. All other hyperparameters are kept consistent with those used in the ShapeNet experiments. During fine-tuning, we strictly follow the 3DETR-m setup to ensure a fair comparison. As shown in Tab. 2, the experimental results clearly demonstrate the effectiveness and generalization ability of our method in large-scale 3D scene settings.

C.3. Ablation of Structural Modeling

Unlike the standard Mamba, we use hidden states to model the structural information of point clouds. During the state update and propagation process, we incorporate the relative positions between input points and states into the generation of SSM parameters. This spatial interaction enables each state to selectively focus on and update the features of points within its designated region. As a result, StruMamba3D eliminates the need for serialization strategies to reorder the point cloud.

To further evaluate the impact of our proposed structural modeling and point serialization strategies, we present the experiments in Tab. 3. For the baseline using the standard Mamba block, adding serialization improves model performance. However, due to the distortion of spatial adjacency, the performance still lags behind our method (89.87 vs. 92.75 on ScanObjectNN, 93.68 vs. 95.06 on ModelNet40, and 84.07 vs. 84.96 on ShapeNetPart). StruMamba3D achieves promising performance without the need for serialization, and adding serialization provides only a modest performance gain. Given the computational overhead introduced by serialization, we choose to remove it from StruMamba3D.

C.4. Ablation of the Number of Spatial States

In this work, we introduce spatial states into the SSM to capture the structural information of point clouds. To investigate the impact of the number of spatial states on model performance, we conduct ablation studies as shown in Sec. C.7.

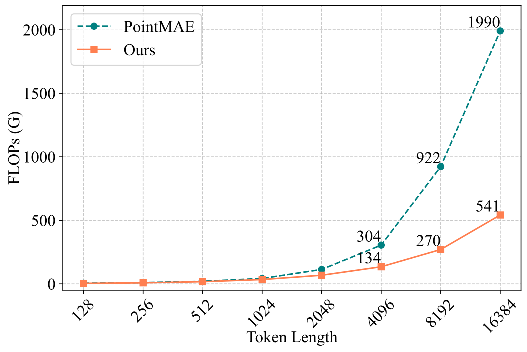

Figure 1. **FLOPs comparison with PointMAE.**

The results show that reducing the number of spatial states leads to a performance drop, highlighting the importance of spatial states in capturing point cloud structures. When the number of spatial states exceeds 16, the performance no longer improves significantly with further increases. We believe that 16 spatial states are sufficient to represent the structural information of objects. Therefore, we set M=16 as the number of spatial states to balance model performance and computational cost.

C.5. Ablation on the Lightweight Convolution

In the structural SSM block, we propose a lightweight convolution for spatial states and input points. This design offers two key benefits: (1) Enhance feature interactions among spatial states. (2) Replace the causal conv1d in the original SSM block. In lightweight convolution, the number of neighboring points determines the receptive field of the module. To select an appropriate number of neighbors, we conduct an ablation study on the number of neighbors for spatial states and input points. As shown in Tab. 5, the model achieves optimal performance when k=4 for spatial states and k=8 for input points. Additionally, we observe that the model is more sensitive to the number of neighbors of spatial states than to that of input points. This observation suggests that spatial states play a more crucial role in capturing structural information.

C.6. Linear Complexity of StruMamba3D

Leveraging hardware-aware scan algorithm of Mamba [3], StruMamba3D exhibits linear computational complexity. As illustrated in Fig. 1, we present a FLOPs comparison with the transformer-based method PointMAE, where StruMamba3D demonstrates significantly lower computational cost when processing long sequences. We also report the inference time and FPS under different input lengths. As shown in Sec. C.6, our method is faster than the Transformer-based method PointGPT, especially with longer input lengths. Besides, our method also achieves competitive inference speed compared to other Mamba-based methods.

Table 1. **Detailed Results of Part Segmentation on ShapeNetPart Dataset.** We report the mean IoU for each category. * indicates results reproduced using the official code.

Method	mIoU$_c$	mIoU$_i$	airplane	bag	cap	car	chair	earphone	guitar
MaskPoint [6]	<u>84.6</u>	86.0	84.2	85.6	88.1	80.3	91.2	79.5	91.9
PointBERT [11]	84.1	85.6	84.3	84.8	88.0	79.8	91.0	**81.7**	91.6
PointMAE [11]	84.2	<u>86.1</u>	84.3	85.0	88.3	<u>80.5</u>	<u>91.3</u>	78.5	<u>92.1</u>
PointGPT-S* [2]	84.0	85.9	**85.1**	<u>86.1</u>	<u>88.8</u>	80.2	91.2	78.8	91.7
PointMamba* [5]	83.8	85.9	84.5	84.7	87.8	80.2	91.2	78.7	91.9
Ours	**85.0**	**86.7**	<u>84.8</u>	**87.4**	**89.2**	**81.3**	**91.6**	<u>80.8</u>	**92.2**

Method	knife	lamp	laptop	motorbike	mug	pistol	rocket	skateboard	table
MaskPoint [6]	87.8	**86.2**	95.3	<u>76.9</u>	<u>95.0</u>	<u>85.3</u>	**64.4**	<u>76.9</u>	81.8
PointBERT [11]	<u>87.9</u>	85.2	95.6	75.6	94.7	84.3	63.4	76.3	81.5
PointMAE [11]	87.4	<u>86.1</u>	**96.1**	75.2	94.6	84.7	<u>63.5</u>	**77.1**	**82.4**
PointGPT-S* [2]	87.0	84.5	<u>95.9</u>	74.2	<u>95.0</u>	83.8	62.6	76.8	81.6
PointMamba* [5]	87.3	85.1	95.8	74.0	94.6	84.7	61.6	75.9	**82.4**
Ours	**88.3**	86.0	95.8	**78.1**	**95.5**	**85.8**	<u>63.5</u>	<u>76.9</u>	<u>82.2</u>

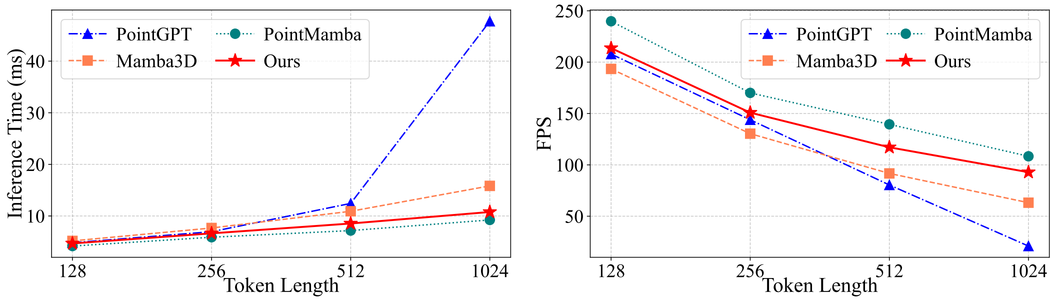

Figure 2. **Inference time and FPS comparison.**

Table 2. **Quantitative comparison on the ScanNetV2 dataset in terms of mAP25 and mAP50.**

Method	mAP$_{25}$	mAP$_{50}$
3DETR	62.1	37.9
3DETR-m	65.0	47.0
Point-M2AE	66.3	48.3
Ours	**67.8**	**50.6**

Table 3. **Ablation of Structural Modeling.**

Backbone	Serialization	Overall Accuracy		mIoU$_c$
		ScanNN	MN40	SNPart
Mamba	✗	88.24	92.50	82.08
Mamba	Z-order	89.28	93.44	83.89
Mamba	Hilbert	89.87	93.68	84.07
StruMamba3D	✗	<u>92.75</u>	**95.06**	<u>84.96</u>
StruMamba3D	Z-order	92.47	<u>94.98</u>	84.86
StruMamba3D	Hilbert	**92.81**	94.89	**85.12**

Table 4. **Ablation of the number M of spatial states.**

M	Overall Accuracy		mIoU$_c$
	ScanNN	MN40	SNPart
4	91.74	93.35	84.34
8	92.26	<u>94.89</u>	84.71
16	**92.75**	**95.06**	<u>84.96</u>
24	<u>92.57</u>	94.81	**85.07**

C.7. Performance without Pretraining

We provide the results of training our model from scratch (without pretraining). As shown in following table, our method still performs well, outperforming existing supervised learning only methods.

Table 5. **Ablation on the point convolution.** "k" denotes the number of k-nearest neighbors in the lightweight convolution.

k for F_h	k for F_x	Overall Accuracy		mIoU$_c$
		ScanNN	MN40	SNPart
4	8	**92.75**	**95.06**	**84.96**
2	8	91.91	94.29	84.54
8	8	92.71	94.98	84.92
4	4	92.30	94.65	84.71
4	12	92.37	94.69	84.86

Method	ScanObjectNN	ModelNet40	ShapeNetPart
	mOA	mOA	mIoU$_c$
w/o Pretraining	91.33	93.68	83.96
MPM Pretraining	92.09	94.45	84.49
Our Pretraining	**92.75**	**95.06**	**84.96**

D. Qualitative Analysis

D.1. Visualization of Spatial State Correlation

We introduce spatial states into the SSM to capture the local structure of point clouds, thereby preserving the spatial dependencies among points. To validate the effectiveness of spatial states, we visualize the correlations between the output point features and spatial state features produced by the structural SSM. Since neighboring spatial states may focus on the same region, we select five spatial states that are widely spaced for visualization. As shown in Fig. 3, different spatial states focus on different parts of the point cloud. The results demonstrate that our structural SSM can effectively capture the local structure of point clouds by spatial states.

D.2. Visualization of Masked Point Modeling

In the pretraining phase, we adopt masked point modeling as our primary self-supervised learning objective. Given a point cloud, we randomly mask 60% of the point groups and use StruMamba3D to extract features, followed by a shallower decoder that predicts the coordinates of masked points. To further analyze the ability of our model to perceive structural information, we visualize the reconstruction results on the ShapeNet dataset. As shown in Fig. 4, our model can effectively reconstruct the original structure even when 60% of the points are masked. Moreover, our model performs remarkably in reconstructing complex geometric patterns, such as the delicate shapes of chair backs and table legs. These visualization results validate that our approach can effectively capture and encode both local and global structural information from point clouds, which is crucial for downstream tasks.

D.3. Visualization of Part Segmentation

In this subsection, we present the qualitative results of part segmentation on the ShapeNetPart validation set, comparing ground truth and predictions. As shown in Fig. 5, our method demonstrates highly competitive performance in part segmentation. Notably, the ShapeNetPart dataset contains certain annotation errors, as highlighted by the red circles. Nevertheless, our method not only achieves a high mIoU but also correctly segments points with erroneous annotations. Moreover, for complex structures such as wheels and chair legs, our approach delivers accurate and reliable segmentation results.

E. Discussions

In this section, we discuss the limitations of our work and potential directions for future research. Our primary goal is to fully exploit the potential of Mamba for point cloud representation learning. To address two key issues of Mamba: disrupting the spatial adjacency of point clouds and struggling to maintain long-sequence memory in downstream tasks, we propose the structural SSM block and the sequence length-adaptive strategy, respectively. While our approach effectively models the structural information of point clouds and achieves significant performance improvements across four downstream tasks, certain challenges remain. Specifically, in part segmentation tasks, we observe some results with unclear boundaries. We attribute this limitation to the nature of single-scale models, which focus on fixed-scale features and struggle to capture geometric details across multiple scales. For example, larger structures such as chair backs and finer details like table legs require different scale features for precise differentiation. To address this, a promising future direction is the development of a hierarchical Mamba that can perform multi-scale point cloud feature extraction through spatial state modeling. This could enhance the ability of the model to adaptively capture both global structures and fine-grained details, leading to more accurate segmentation results.

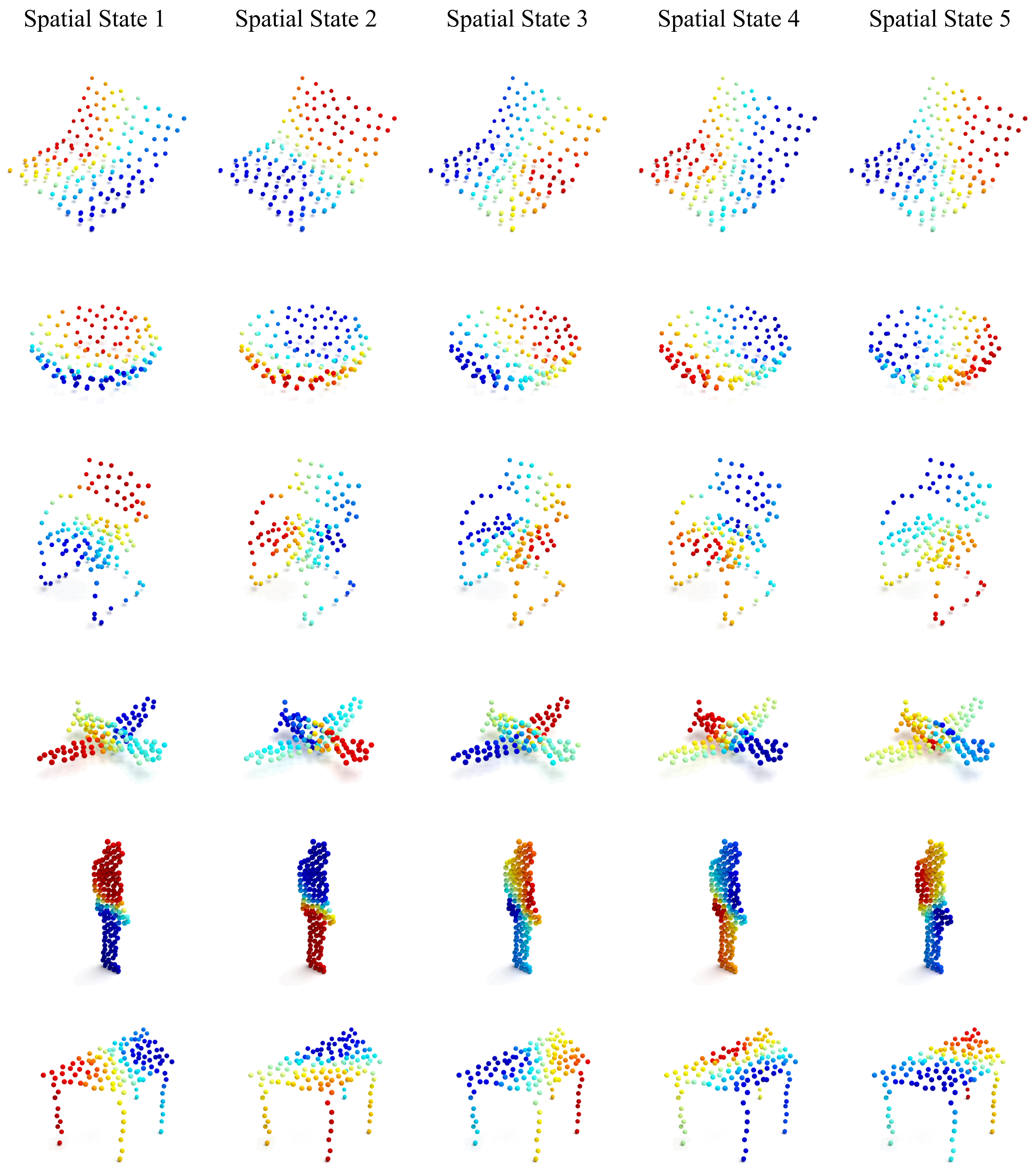

Figure 3. **Visualization of correlation between the output point and spatial state features produced by the structural SSM.** For each point cloud, we select five spatial states that are widely spaced for visualization.

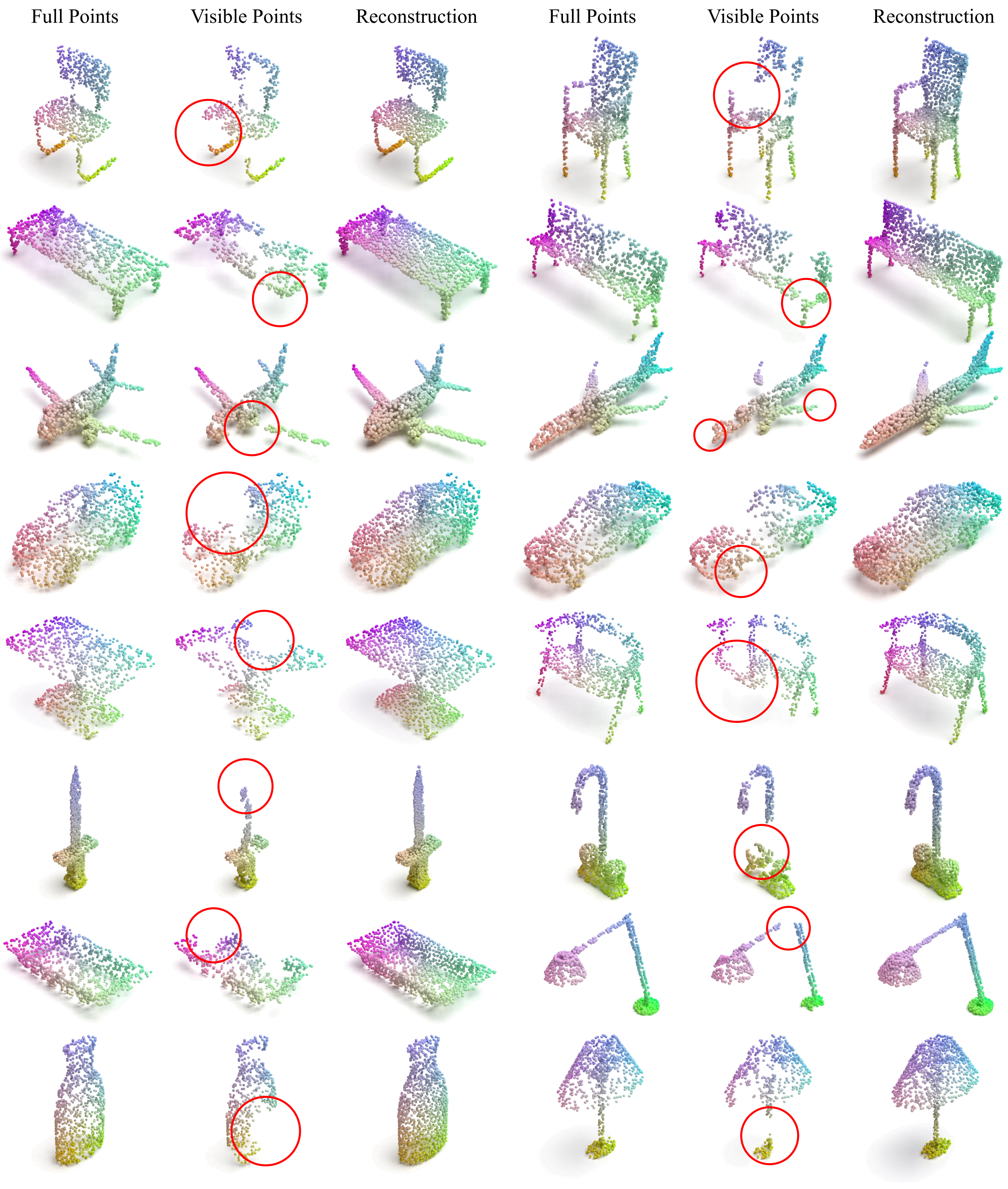

Figure 4. **Visualization of reconstructed masked regions on ShapeNet.** Full points represent the raw point cloud, while visible points correspond to the input points with 60% masked. The reconstruction results consist of the predicted points combined with visible points.

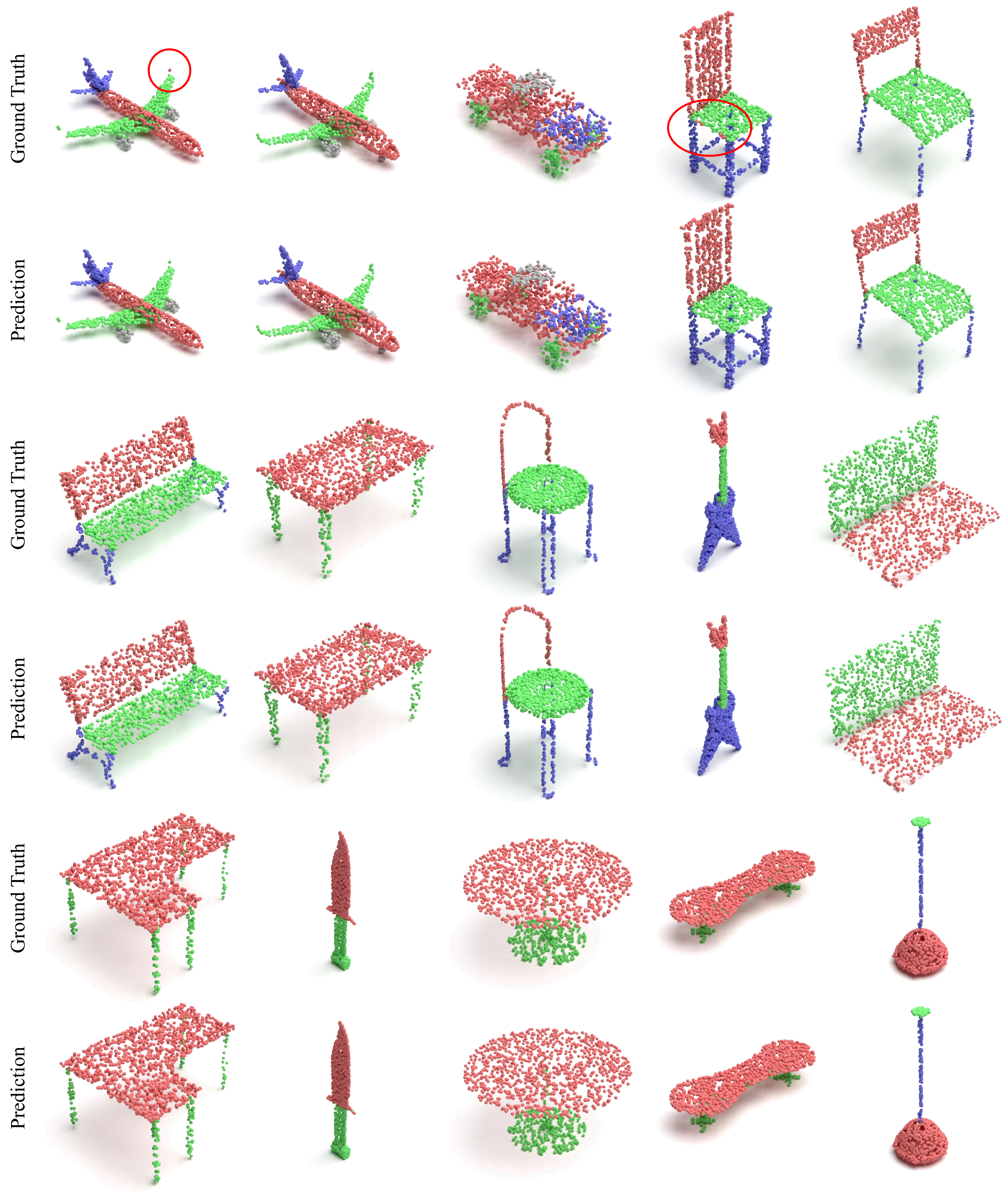

Figure 5. **Visualization of part segmentation of our Structural Mamba on ShapeNetPart.** Different colors represent different parts. The red circles highlight points with obvious annotation errors.

References

[1] 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. 1

[2] Guangyan Chen, Meiling Wang, Yi Yang, Kai Yu, Li Yuan, and Yufeng Yue. Pointgpt: Auto-regressively generative pre-training from point clouds. *arXiv preprint arXiv:2305.11487*, 2023. 2, 4

[3] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 3

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1, 2

[5] Dingkang Liang, Xin Zhou, Wei Xu, Xingkui Zhu, Zhikang Zou, Xiaoqing Ye, Xiao Tan, and Xiang Bai. Pointmamba: A simple state space model for point cloud analysis. In *Advances in Neural Information Processing Systems*, 2024. 2, 4

[6] Haotian Liu, Mu Cai, and Yong Jae Lee. Masked discrimination for self-supervised learning on point clouds. In *European Conference on Computer Vision*, pages 657–675. Springer, 2022. 4

[7] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 1, 2

[8] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *European conference on computer vision*, pages 604–621. Springer, 2022. 1

[9] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2

[10] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016. 2

[11] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19313–19322, 2022. 1, 4