Serialization based Point Cloud Oversegmentation

Supplementary Material

A. Implementation details

In this section, we provide more implementation details of the experiments. Our implementation primarily refers to the Pointcept codebase [1]. The train settings are shown in Tab. 1. The model settings are presented in Tab. 2. The data augmentations shown in Tab. 3 are the same as those in PTv3 [2].

A.1. A.Implementation details

In this section, we provide more implementation details of the experiments.

A.1.1. Model Configuration

We employ a consistent architecture across all experiments with minimal dataset-specific adjustments. Our model utilizes a modified sparse U-Net backbone featuring five-stage symmetric encoder-decoder structure, each with a single block depth. The network's embedding dimensions progress through [32, 64, 128, 256] in the encoder and [128, 64, 64] in the decoder, with Mamba blocks incorporating conditional positional encoding at the 256 dimensional stages. For hierarchical point cloud processing, we implement a three-level hierarchy using Hilbert curves, organizing points into n, 1024, 256 segments (where n represents the original point count). The superpoint feature update module maintains a 64-dimensional feature space and iterates three times, matching the raw point embedding dimension. The superpoint graph interaction component comprises four consecutive layers, utilizing a k-NN graph structure with k=6 in feature space for the Graph Transformer and k=3 in Euclidean space for the GCN operations. Training is performed on two NVIDIA RTX 4090 GPUs using Adam optimizer with early stopping when validation recall stagnates for 20 epochs, and the final results are reported as the average across all epochs.

B. Supplemental Explanation

B.1. Hilbert Curve Mapping

Time Complexity. Our Hilbert curve mapping follows the implementation of PTv3 [2], and its time complexity can be divided into several parts. First, the Coordinate Quantization process discretizes each 3D point into a grid of resolution $2^m \times 2^m \times 2^m$, where m is the bit-depth. This requires $\mathcal{O}(1)$ operations per point, as quantization involves fixed-bit truncation. Second, the Hilbert Value Generation maps from 3D coordinates to 1D Hilbert indices through iterative bit-level permutations. For m bits, each point undergoes $\mathcal{O}(m)$ bitwise operations (*e.g.* XOR, shifts) across three dimensions. The total cost for n points is $\Theta(n \times m)$. In practical implementations, m is bounded (*e.g.* $m = 16$), reducing this term to $\mathcal{O}(n)$. Combining quantization, Hilbert

Indoor Config	Indoor Value	Outdoor Config	Outdoor Value
optimizer	AdamW	optimizer	AdamW
scheduler	Cosine	scheduler	Cosine
criteria	CrossEntropy(1) Lovasz(1)	criteria	CrossEntropy(1) Lovasz(1)
learning rate	1e-3	learning rate	8e-4
block lr scaler	0.1	block lr scaler	0.1
weight decay	5e-2	weight decay	5e-2
batch size	8	batch size	8
datasets	ScanNet / S3DIS	datasets	NuScenes / Sem.KITTI
warmup epochs	40	warmup epochs	2
epochs	800	epochs	50

Table 1. The train settings of indoor and outdoor scenes.

SpConv setting	**Value**
embedding channels	64
encoder depth	[1, 1, 1, 1]
encoder channels	[32, 64, 128, 256]
decoder depth	[1, 1, 1]
decoder channels	[64, 64, 128]
down stride	[2, 2, 2]
Mamba setting	**Value**
serialization order	Hilbert order
embedding layers	4^{th}
hidden state	8
expand	1
drop path	0.1
Superpoint setting	**Value**
granularity	[1024, 256]
initial method	averge pooling
number of nei. expansion	2
similarity metric	cosine similarity
superpoint dimension	64
number of iterations(level 1)	3
number of iterations(level 2)	3
loss setting	**Value**
predefined separation margin δ_{dist}	0.5
loss weights δ	1

Table 2. The model settings.

Augmentations	Parameters	Indoor	Outdoor
random dropout	dropout ratio: 0.2, p: 0.2	✓	-
random rotate	axis: z, angle: [-1, 1], p: 0.5	✓	✓
	axis: x, angle: [-1 / 64, 1 / 64], p: 0.5	✓	-
	axis: y, angle: [-1 / 64, 1 / 64], p: 0.5	✓	-
random scale	scale: [0.9, 1.1]	✓	✓
random flip	p: 0.5	✓	✓
random jitter	sigma: 0.005, clip: 0.02	✓	✓
elastic distort	params: [[0.2, 0.4], [0.8, 1.6]]	✓	-
auto contrast	p: 0.2	✓	-
color jitter	std: 0.05; p: 0.95	✓	-
grid sampling	grid size: 0.02 (indoor), 0.05 (outdoor)	✓	✓
sphere crop	ratio: 0.8, max points: 128000	✓	-
normalize color	p: 1	✓	-

Table 3. The data augmentations.

Algorithm 1 Hierarchical Quaternary Partitioning on Serialized Point Cloud

```
Require: Input point cloud $\mathcal{P} \in \mathbb{R}^{N\times 3}$, total levels $K$
Ensure: $K$-level hierarchical structure $\mathcal{H}$
 1: function BuildHierarchy($\mathcal{P}$)
 2: $\mathcal{P}_{seq} \leftarrow \mathcal{H}(\mathcal{P})$ {Hilbert curve serialization}
 3: $\mathcal{P}_{pad} \leftarrow \phi(\mathcal{P}_{seq}, 4^K)$ {Pad to nearest multiple of $4^K$}
 4: return QuaternaryPartition($\mathcal{P}_{pad}$, 0, $N_{pad}$, 1, $K$)
 5: end function
 6: function QuaternaryPartition($\mathcal{P}_{pad}$, $start$, $end$, $k$, $K$)
 7: $S_i^k \leftarrow \{\mathcal{P}_{pad}[j] \mid j \in [start, end)\}$
 8: if $k = K$ then
 9:     return $S_i^k$
10: end if
11: $step \leftarrow (end - start)/4$
12: $mid_1 \leftarrow start + step$
13: $mid_2 \leftarrow start + 2 \times step$
14: $mid_3 \leftarrow start + 3 \times step$
15: $S_{4i}^{k+1} \leftarrow$ QuaternaryPartition($\mathcal{P}_{pad}$, $start$, $mid_1$, $k+1$, $K$)
16: $S_{4i+1}^{k+1} \leftarrow$ QuaternaryPartition($\mathcal{P}_{pad}$, $mid_1$, $mid_2$, $k+1$, $K$)
17: $S_{4i+2}^{k+1} \leftarrow$ QuaternaryPartition($\mathcal{P}_{pad}$, $mid_2$, $mid_3$, $k+1$, $K$)
18: $S_{4i+3}^{k+1} \leftarrow$ QuaternaryPartition($\mathcal{P}_{pad}$, $mid_3$, $end$, $k+1$, $K$)
19: $\mathcal{I}_i^k \leftarrow \{4i, 4i+1, 4i+2, 4i+3\}$ {Store quad-child indices}
20: return $S_i^k \cup \{S_{4i}^{k+1}, S_{4i+1}^{k+1}, S_{4i+2}^{k+1}, S_{4i+3}^{k+1}\}$
21: end function =0
```

mapping, and sorting:

$$T(n) = \mathcal{O}(n) + \mathcal{O}(n) = \mathcal{O}(n). \quad (1)$$

B.2. Hierarchical Segment Initialization

The hierarchical partitioning algorithm for serialized point clouds is detailed in Algorithm 1.

C. Additional Ablation Studies

Since our oversegmentation network can be combined with a semantic segmentation head to form an end-to-end architecture, we directly optimize a joint loss function as shown in Eq. (2):

$$\ell_{joint} = \delta\ell_{sp} + \ell_{sem} \quad (2)$$

where ℓ_{sp} represents the local discriminative loss for training superpoints, and ℓ_{sem} is the cross-entropy loss for semantic segmentation. The parameter δ serves as the weighting factor that controls the influence of the superpoint aggregation loss.

The superpoint loss ℓ_{sp} consists of three components as formulated in Eq. (3):

$$\begin{aligned}\ell_{sp} =& \frac{1}{N_s}\sum_{i=1}^{N_s}\frac{\sum_{p\in S_i}\|f_i - f'_{S_i^k}\|^2}{|S_i|} + \\ & \frac{1}{N_c}\sum_{i,j\in c, i\neq j}\max(0, \delta_{dist} - \|f'_{S_i^k} - f'_{S_j^k}\|)^2 + \\ & \frac{1}{N_s}\sum_{i=1}^{N_s}\sum_{k=1}^{K} p_{i,k}\log(p_{i,k})\end{aligned} \quad (3)$$

To investigate the impact of the superpoint loss weighting, we conducted experiments with various δ values. Fig. 1 presents the Boundary Precision (BP) metric across training epochs for different weight configurations. The results demonstrate that as δ increases from 0 to 1.0, the BP metric maintains relatively high values, with weights of 0.4, 0.5, and 1.0 achieving the best performance (consistently around 30%).

Conversely, when δ gradually decreases, particularly when it approaches or equals zero, we observe a dramatic decline in BP performance, dropping to approximately 20% by the end of training. This trend clearly validates the effectiveness of our proposed superpoint aggregation loss formulated in Eq. (3). The higher weights (0.4-1.0) show stable performance across training epochs, while lower weights (0-0.1) exhibit significant degradation over time.

Notably, a weight of 0.2 maintains a reasonable compromise between stability and performance. These findings suggest that proper balancing of the superpoint loss component is crucial for preserving boundary precision in our joint learning framework.

D. Additional visualization

We further visualized additional superpoints generated by SPCNet and the prediction results on ScanNet and Sem.KITTI, as shown in Fig. 2.

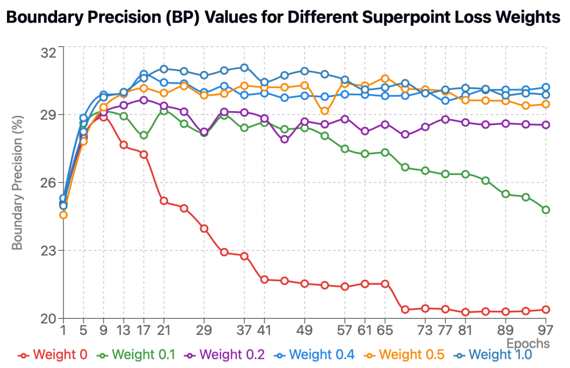

Figure 1. Boundary Precision (BP) values over training epochs for different superpoint loss weights (δ) on the ScanNet.

E. Limitations

Sensitivity to Serialization Patterns: Our voxel serialization relies on 3D Hilbert curves to preserve spatial locality. While effective for regular indoor scenes, this approach may struggle with complex outdoor geometries (*e.g*. overpasses, multi-level vegetation) where Hilbert ordering could disrupt natural spatial adjacency. The partitioning granularity following a geometric progression $M_k = 2^{m-2k+2}$ also limits adaptation to scenes with extreme scale variations. Future work could explore adaptive curve selection and dynamic granularity adjustment based on local geometric complexity.

Task-Specific Feature Coupling: The end-to-end framework jointly optimizes superpoint generation and semantic segmentation, potentially causing overfitting to segmentation objectives at the expense of generalizable superpoint properties. While our ablation studies validate superpoint quality, the lack of dedicated oversegmentation benchmarks necessitates caution—superpoints optimized for one semantic segmentation dataset may not transfer optimally to other tasks like instance segmentation, where instance segmentation datasets would be needed to optimize superpoints, and superpoint boundaries would then conform to instance object boundaries. Decoupled training protocols or multi-task losses could mitigate this limitation.

F. Future Work

While the current superpoint refinement method efficiently uses hard assignments based on similarity matching, it risks propagating early errors through misassigned heterogeneous regions. To mitigate this, our framework could integrate two complementary improvements: First, adaptive region-growing along the serialized curve could dynamically merge clusters by expanding from high-confidence seeds until encountering feature divergence thresholds, preserving spatial coherence. Second, semantic-consistency correction could reassign outlier points by leveraging the observed class dominance within superpoints, statistically reallocating minority-class points to adjacent segments with compatible profiles while respecting hierarchical adjacency constraints. These strategies synergistically balance boundary awareness with topological consistency.

References

[1] Pointcept Contributors. Pointcept: A codebase for point cloud perception research. https://github.com/Pointcept/Pointcept, 2023. 1

[2] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4840–4851, 2024. 1

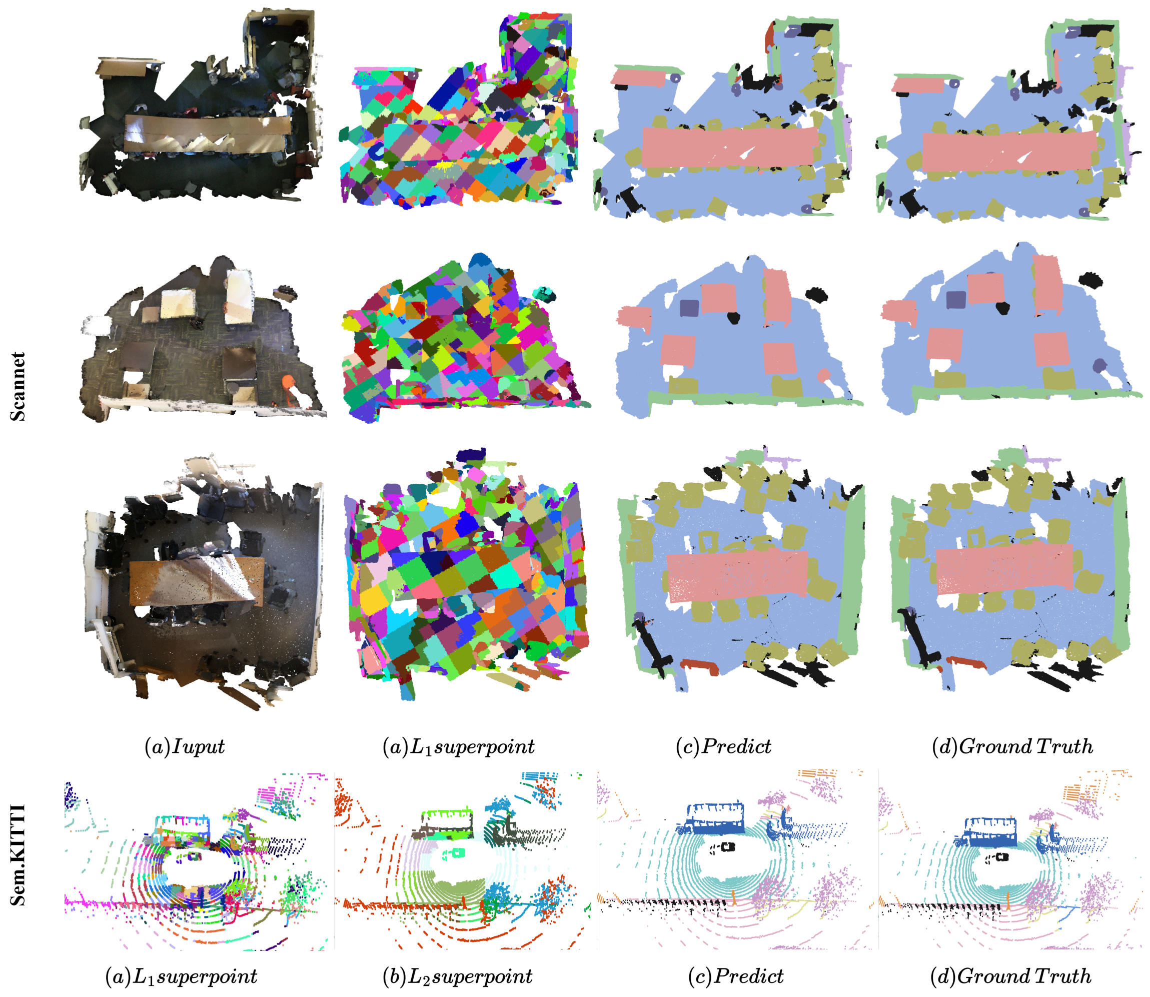

Figure 2. The visualization of superpoints generated on the ScanNet and Sem.KITTI.