2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

EZ-SP: Fast and Lightweight Superpoint-Based 3D Segmentation

Louis Geist[1] Loic Landrieu[1] Damien Robert[2]

[1] LIGM, ENPC, IP Paris, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France
[2] DM3L, University of Zurich, Zurich, Switzerland

***Abstract*— Superpoint-based pipelines provide an efficient alternative to point- or voxel-based 3D semantic segmentation, but are often bottlenecked by their CPU-bound partition step. We propose a learnable, fully GPU partitioning algorithm that generates geometrically and semantically coherent superpoints 13× faster than prior methods. Our module is compact (under 60k parameters), trains in under 20 minutes with a differentiable surrogate loss, and requires no handcrafted features. Combined with a lightweight superpoint classifier, the full pipeline fits in <2 MB of VRAM, scales to multi-million-point scenes, and supports real-time inference. With 72× faster inference and 120× fewer parameters, EZ-SP matches the accuracy of point-based SOTA models across three domains: indoor scans (S3DIS), autonomous driving (KITTI-360), and aerial LiDAR (DALES). Code and pretrained models are accessible at `github.com/drprojects/superpoint_transformer`.**

I. Introduction

Accurate 3D semantic segmentation is critical for robotic perception, enabling tasks such as autonomous driving [1], navigation [2], [3], and mapping [4]. However, balancing accuracy with computational efficiency remains a major challenge. Real-world point clouds often contain millions of points and must be processed under strict latency constraints—for instance, rotating automotive LiDAR typically acquires 1.3M points per second. Yet, state-of-the-art (SOTA) models routinely exceed ten million parameters, rely on costly test-time augmentation to reach their performance, and can only process small scenes at a time. This gap limits deployment in real-time robotic systems, as well as in AR/VR [5] and large-scale smart city applications [6].

A common strategy to reduce complexity is to group points into *superpoints* [7], [8]: spatially contiguous, geometrically homogeneous regions. Reasoning on superpoints rather than individual points dramatically reduces memory and computation, while maintaining SOTA or near-SOTA accuracy [9], [10]. However, the partition stage remains a critical bottleneck: often CPU-bound, slow to tune (each parameter sweep may take hours), and requires complex optimization over handcrafted geometric and radiometric features. This overhead in runtime and engineering effort has hampered the broader adoption of superpoint-based methods.

To overcome these limitations, we introduce EZ-SP (easy-superpoints): a lightweight, *learnable, fully GPU* pipeline to partition and segment raw point clouds into superpoints *on the fly*. A 60k-parameter backbone produces low-dimensional embeddings optimized to detect *semantic transitions* directly from raw point clouds and without handcrafted features. We then use a massively parallel clustering algorithm to produce semantically homogeneous, geometrically simple superpoints—124× faster than the widely used Parallel Cut Pursuit [11], [12]. These superpoints are then processed by a 330k-parameter superpoint segmentation network, yielding dense labels at a fraction of the cost of point- or voxel-based approaches. As shown in fig. 1, our method matches or surpasses SOTA accuracy while delivering over 72× faster end-to-end inference than PointTransformer-v3 [13].

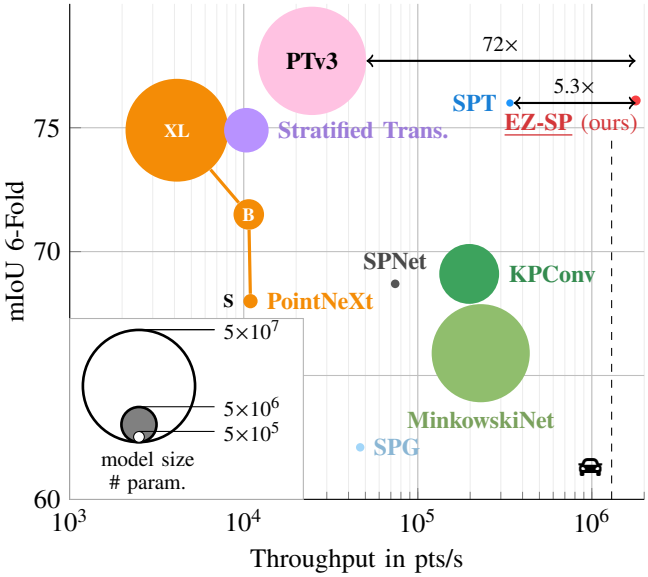


Fig. 1: **Inference Speed *v.s.* Performance *v.s.* Model Size.** Comparison of end-to-end pipelines (preprocessing to inference) on S3DIS. EZ-SP achieves near-SOTA accuracy with only 400k parameters, while being orders of magnitude faster than point-based networks, and the only method to match the acquisition rate of automotive LiDAR (🚘).

Key advantages and contributions. Our approach is:

- **Fast**: partitioning a raw point cloud is over 13× faster than the fastest graph-based approaches; end-to-end inference is over 72× faster than recent SOTA models.
- **Lightweight**: the entire model fits in <2 MB VRAM and trains from scratch in 3 h on a single A6000.
- **Versatile**: The same configuration generalizes across indoor (S3DIS), mobile (KITTI-360), and aerial mapping (DALES) with minimal hyperparameter tuning. Our model runs in real time on small scans (>1.3M pt/s) and scales to city-scale point clouds in a single pass.
- **Scalable**: GPU-only segmentation of tens of millions of points in a single pass on a single consumer-grade GPU.

979-8-3315-8160-2/26/$31.00 ©2026 IEEE

II. Related Work

This section presents an overview of 3D deep learning with a focus on superpoint-based approaches.

3D Semantic Segmentation. Most existing approaches operate directly on raw points or on fine voxel grids, leveraging convolutional architectures [14], [15], [16] or, more recently, transformers [17], [18], [19], [20], [13]. These methods achieve state-of-the-art accuracy, but at substantial computational cost. Models typically contain tens of millions of parameters and require significant memory and processing time, limiting their applicability to large-scale point clouds and real-time robotic systems.

Superpoint-Based Semantic Segmentation. Partitioning a scene into *superpoints*—spatially contiguous, geometrically homogeneous point clusters—drastically reduces computational load by shifting semantic reasoning from points to regions. The *Superpoint Graph* (SPG) [7] pioneered this approach, achieving competitive results with far fewer resources. The *Superpoint Transformer* (SPT) [10] advanced the idea with hierarchical partitions and sparse self-attention, reaching SOTA accuracy with only ~200k parameters. Extensions include instance [21], [22], panoptic [23], weakly supervised [24], and self-supervised [25], [26], [27] segmentation. However, all rely on CPU-based, handcrafted partitions, which limit scalability and accuracy.

Classical Superpoint Partitioning. Early 3D oversegmentation methods often used supervoxels, *e.g.* , VCCS [8] via voxel-based k-means. Such methods require *predefining the number of clusters* and are sensitive to initialization, preventing flexibility to scene size and local complexity. Saliency-guided [28] and density-adaptive [29] variants improved boundaries, while graph-based optimization [30], [31], [21], [32], [33] further improved adaptability. Nevertheless, these methods remain CPU-bound and largely depend on handcrafted features and hyperparameters. Sparsification methods [34], [35], [36] lower the number of points but do not produce valid partitions for segmentation. Image-based methods [37] typically project SLIC [38] or SAM [39] superpixel partitions onto 3D points, but require co-registered 2D images, and rely on 2D textures rather than 3D geometry.

Learning Superpoint Partitions. Learning partitions is challenging due to the non-differentiability of standard clustering. GPU-friendly k-means [40], [9], [25], [41] and region-offset [42] approaches enable differentiability but inherit k-means' limitations and often require heavy encoders. GraphCut [43] introduces a graph-based heuristic on learned embeddings for instance segmentation, while SAM-Graph [44] transfers priors from large image models to generate superpoints for detection. Supervised SuperPoints (SSP) [45] directly train embeddings to highlight boundaries but still rely on CPU-based Cut Pursuit [12]. Our method replaces the CPU-based solver with a massively parallel approximate algorithm, uses an edge-based surrogate loss, and employs a *small* (330k-parameter) encoder. In practice, removing the CPU bottleneck and avoiding fixed cluster counts enable fast, scalable training and inference.

III. Method

We consider a point cloud $\mathbf{P}$ composed of points with F features: spatial coordinates and potential radiometric attributes (color, intensity). Each point p has a semantic label $\text{cls}(p) \in [1, C]$, with C classes. Our goal is to efficiently predict the semantic labels of all points. We first learn low-dimensional embeddings tailored for detecting semantic transitions (section III-A), then propose an efficient GPU-based superpoint partitioning method (section III-B), and finally show how our approach can be interfaced with a superpoint classification model for fast end-to-end semantic segmentation (section III-C). See fig. 2 for a visual overview.

A. Detecting Semantic Transition

We train a lightweight network to compute point embeddings optimized for detecting semantic transitions.

Motivation. Directly embedding points in a semantic space is typically challenging and requires large networks with extensive receptive fields. Instead, we exploit the simpler insight that semantic boundaries usually correspond to sharp contrasts in geometry or radiometry, such as the geometric discontinuity between a chair and the floor, or the change of color between doors and walls. Detecting these *semantic transitions* is therefore a much simpler problem than semantic segmentation, does not require global information, and should be achievable with a simpler model.

Point Embedding. We define the embedding function $\phi^{\text{point}} : \mathbb{R}^{|\mathbf{P}|\times F} \mapsto \mathbb{R}^{|\mathbf{P}|\times M}$, which associates each point p with an M-dimensional embedding vector. We denote by $\mathbf{X} = \phi^{\text{point}}(\mathbf{P})$ the embedding matrix.

Semantic Transition Prediction. We aim to learn embeddings that remain homogeneous within objects while being sharply contrasted across semantic boundaries. Following Robert *et. al.* [23], we formulate transition detection as a binary edge classification task. We first define the pairwise affinity between two points p and q as:

$$a_{p,q} = \exp\left(-\|\mathbf{X}_p - \mathbf{X}_q\|/\tau\right) \ , \tag{1}$$

with $\tau > 0$ a temperature parameter. We construct an undirected graph $(\mathbf{P}, \mathbf{E})$ connecting points with their k nearest neighbours. We define *intra-edges*, $\mathbf{E}_{\text{intra}} = \{(p,q) \in \mathbf{E} \mid \text{cls}(p) = \text{cls}(q)\}$, and *inter-edges*, $\mathbf{E}_{\text{inter}} = \{(p,q) \in \mathbf{E} \mid \text{cls}(p) \neq \text{cls}(q)\}$. We encourage $a_{p,q} \approx 1$ for intra-edges and $a_{p,q} \approx 0$ for inter-edges with a *contrastive loss*:

$$\mathcal{L} = \sum_{(p,q)\in\mathbf{E}_{\text{intra}}} -\log(a_{p,q}) + \sum_{(p,q)\in\mathbf{E}_{\text{inter}}} -\log(1 - a_{p,q}) \ . \tag{2}$$

To improve diversity in the learned embeddings, we apply adaptive sampling by randomly dropping intra-edges until they constitute at most ρ_{intra} of the sampled edges.

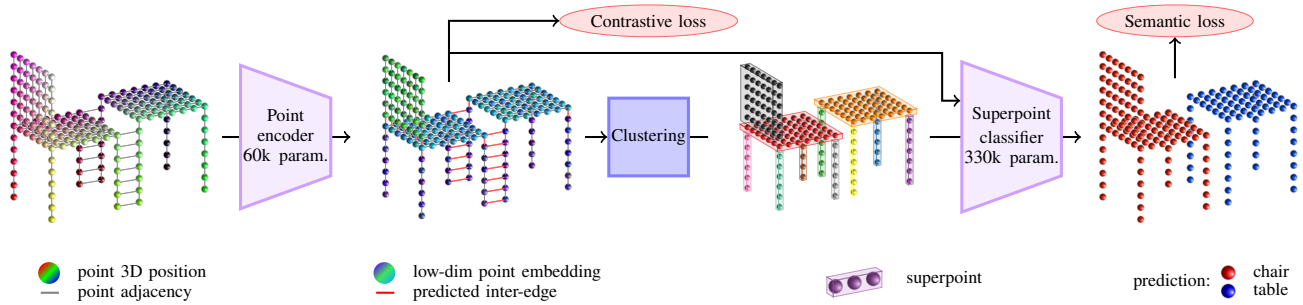


Fig. 2: **EZ-SP.** A 60k-parameter backbone embeds every point of the input scene into a low-dimensional space where adjacent points from different semantic classes (inter-edges) are pushed apart. A GPU-accelerated algorithm then clusters neighbouring points with similar embeddings, producing semantically homogeneous superpoints. Finally, a lightweight (330k-parameter) superpoint-level network assigns a label to each superpoint, which is broadcast back to its points for dense segmentation.

B. Fast Superpoint Partitioning

We now present a GPU-accelerated method to efficiently compute a superpoint partition from the learned embeddings.

Motivation. Clustering-based approaches such as K-means require a fixed cluster count, and thus struggle with variable scene size and complexity. We adopt a graph-based partition approach by minimizing the following contour-regularized energy [12]:

$$\arg\min_{\mathbf{Y}\in\mathbb{R}^{|\mathbf{P}|\times M}} \Omega(\mathbf{Y};\mathbf{X},\mathbf{E}) \text{ with} \tag{3}$$

$$\Omega(\mathbf{Y};\mathbf{X},\mathbf{E}) = \sum_{p\in\mathbf{P}} \|\mathbf{X}_p - \mathbf{Y}_p\|^2 + \lambda \sum_{(p,q)\in\mathbf{E}} w_{p,q}\|\mathbf{Y}_p - \mathbf{Y}_q\|_0 \ ,$$

where $\|x\|_0 = 0$ if $x = 0$ or otherwise 1; $w_{p,q} > 0$ are edge weights; $\lambda > 0$ the regularization strength. Minimizing this energy produces a piecewise-constant approximation $\mathbf{Y}$ of the embeddings $\mathbf{X}$, whose components form our superpoints. As $\mathbf{X}$ is trained to be homogeneous within objects and contrasted at their interface, the superpoints should be semantically coherent. Existing solvers for eq. (3) are typically CPU-bound [11], which can become computational bottlenecks.

Combinatorial Clustering. We recast the non-continuous, non-convex optimization problem of eq. (3) as a combinatorial problem which we can efficiently approximate with a parallel greedy bottom-up merging strategy. Let $\mathcal{P}$ denote a partition of $\mathbf{P}$ into superpoints: each superpoint $P \in \mathcal{P}$ defines a connected component of the graph $(\mathbf{P},\mathbf{E})$ and $\cup_{P\in\mathcal{P}} = \mathbf{P}$. We define the adjacency between superpoints as follows:

$$\mathcal{E} = \{(P,Q) \in \mathcal{P}^2 \mid \exists (p,q) \in \mathbf{E}, p \in P, q \in Q\} \ . \tag{4}$$

We associate each superpoint P with its mean embedding:

$$\mathbf{X}_P = \frac{1}{|P|}\sum_{p\in P} \mathbf{X}_p \ . \tag{5}$$

We then define the point embedding matrix $\mathbf{X}^{\mathcal{P}}$ in $\mathbb{R}^{|\mathbf{P}|\times M}$ which associates each point p with the value $\mathbf{X}_P$ of the superpoint P it belongs to:

$$\mathbf{X}^{\mathcal{P}}_p = \mathbf{X}_P \text{ for } P \text{ such that } p \in P \ . \tag{6}$$

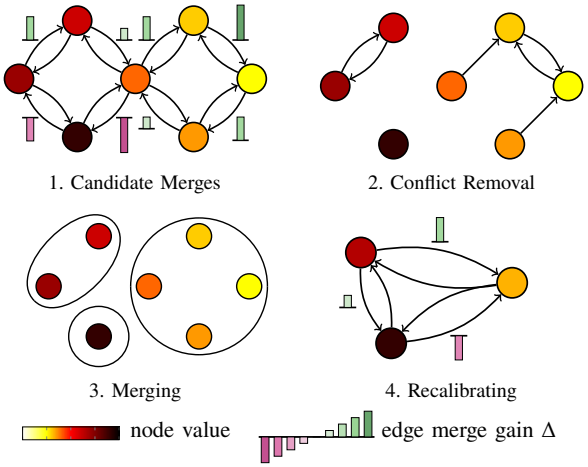


Fig. 3: **Parallel Combinatorial Partition.** Our algorithm greedily approximates a graph signal with piecewise-constant components. Conflicting merges (nodes with multiple outgoing edges) are removed, enabling an efficient parallel implementation on GPUs.

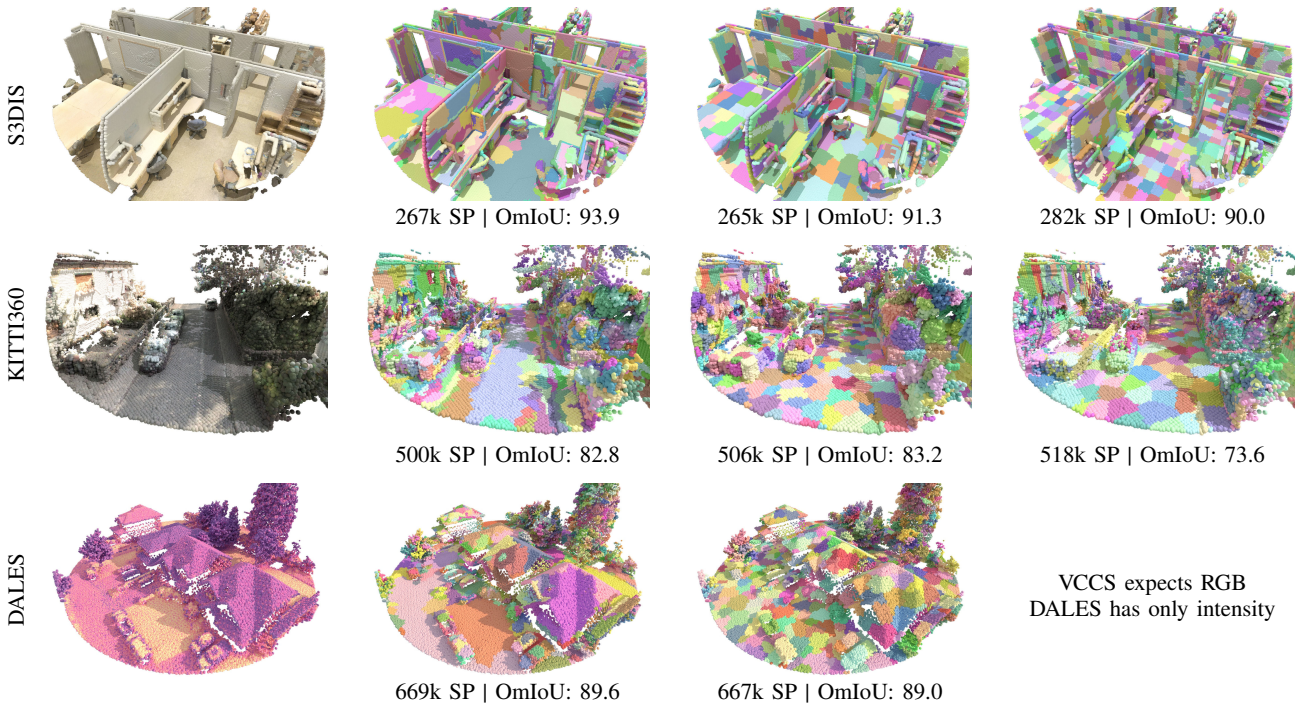
This allows us to restate eq. (3) as a combinatorial problem of minimizing $\Omega(\mathcal{P}) = \Omega(\mathbf{X}^{\mathcal{P}};\mathbf{X},\mathbf{E})$ with respect to $\mathcal{P}$. To do so, we use the following proposition:

Proposition 1. *Merging adjacent superpoints* $(P,Q) \in \mathcal{E}$ *decreases* $\Omega(\mathcal{P})$ *by the following* edge merge gain:

$$\Delta(P,Q) = -\frac{|P||Q|}{|P|+|Q|}\|\mathbf{X}_P - \mathbf{X}_Q\|^2 + \lambda \sum_{(p,q)\in(P\times Q)\cap\mathbf{E}} w_{p,q} \ . \tag{7}$$

Parallel Implementation. The combinatorial problem defined above can be approximated with a greedy merging strategy: at each step, adjacent superpoints (P,Q) with the energy gain $\Delta(P,Q)$ are merged. This process is inherently sequential, since each merge alters subsequent gains, making naive approaches ill-suited for GPUs. We therefore propose a bottom-up, GPU-parallel algorithm, illustrated in fig. 3:

0. **Initialization.** Each point is its own superpoint: $\mathcal{P} = \{\{p\} \mid p \in \mathbf{P}\}$ with adjacency $\mathcal{E} = \mathbf{E}$.
1. **Candidate Merges.** We construct $\mathcal{E}_{\text{merge}}$ the set of *directed* edges $(P \to Q)$ for $(P,Q) \in \mathcal{E}$ satisfying



(a) Input (b) EZ-SP (c) PCP (d) VCCS

Fig. 4: **Partition Examples.** Visualization of point cloud partitions across three datasets and three partitioning algorithms. Section IV-A shows the full dataset sizes; we also report, for each configuration, the resulting number of superpoints and the partition purity over the validation dataset (all folds for S3DIS).

either $\Delta(P, Q) > 0$ or $|P| < \sigma_{\min}$, where $\sigma_{\min}$ is the minimum superpoint size. If $\mathcal{E}_{\text{merge}}$ is empty, return $\mathcal{P}$.

2. **Conflict Removal.** To prevent conflicting merges, each superpoint may appear at most once as a source in $\mathcal{E}_{\text{merge}}$. For each P, we retain only the outgoing edge $(P \to \cdot)$ with the highest merge gain Δ.
3. **Merging.** The remaining edges $\mathcal{E}_{\text{merge}}$ define a directed merge graph $(\mathcal{P}, \mathcal{E}_{\text{merge}})$. We compute its weakly connected components and update $\mathcal{P}$ with the resulting merged sets. This allows chain merges (*e.g.* $(P \to R)$ and $(Q \to R)$) to be resolved in a single iteration.
4. **Recalibration.** Update the node embeddings and merge gains for the new adjacency graph, then return to Step 1.

Our approach makes heavy use of the highly optimized `scatter` operation [46], enabling efficient GPU parallelization. Detailed pseudo-code and proofs of correctness will be released alongside an open-source GPU implementation.

Hierarchical Partition. The proposed algorithm can be applied recursively to produce a *hierarchical* set of partitions, *i.e.* $\mathcal{P}^{(1)}, \ldots, \mathcal{P}^{(L)}$, where $\mathcal{P}^{(1)}$ is a partition of $\mathbf{P}$ and $\mathcal{P}^{(l+1)}$ is a partition of $\mathcal{P}^{(l)}$. This is straightforward to implement by maintaining the adjacency graph between components at each stage. Such multi-scale partitioning is useful for downstream processing, as discussed in the next section.

C. Semantic Segmentation

Once the initial point cloud $\mathbf{P}$ is partitioned into superpoints $\mathcal{P}$, we can apply a superpoint-based classifier to predict their semantic labels. Labels are then broadcast from superpoints back to their constituent points, allowing the inference stage to operate entirely on the much smaller set $\mathcal{P}$ while still producing dense predictions over $\mathbf{P}$.

Superpoint Classification. For classification, we employ the SuperPoint Transformer (SPT) [10] due to its strong balance between accuracy and efficiency. We retain the default configuration with three key modifications:

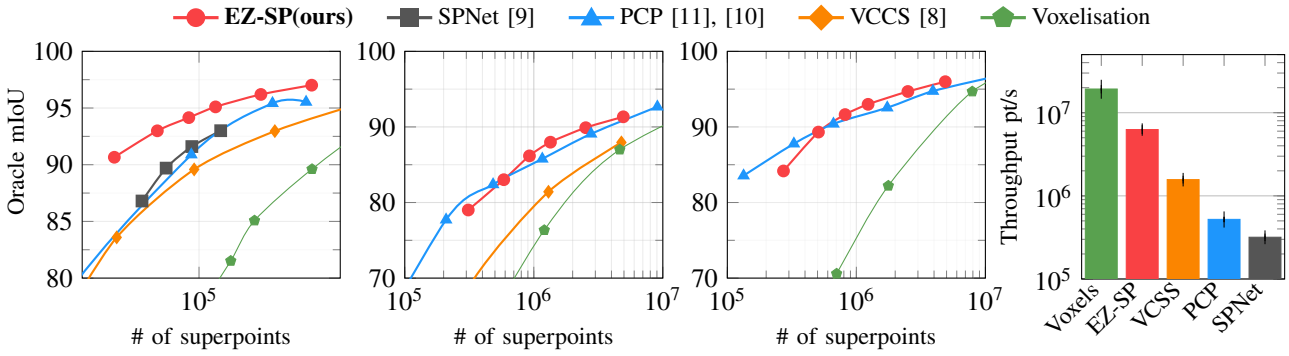
- **Simplified Hyperparameters:** We remove all CPU-bound preprocessing and their hyperparameters. Partition coarseness is controlled solely by one parameter per partition level: the minimum superpoint size.
- **Efficient GPU Pipeline:** because partitioning is fully GPU-based, vectorized operations run faster, and costly CPU–GPU data transfers are eliminated or optimized.
- **Hierarchical Architecture:** we preserve most of the original SPT design but extend it to three nested partition levels to leverage our hierarchical superpoints.

IV. Experiments

We evaluate EZ-SP on three large-scale 3D segmentation benchmarks. We first outline the experimental setup (section IV-A), then assess partition quality and efficiency (section IV-B). Next, we report semantic segmentation performance with a downstream superpoint classifier (section IV-C), followed by an ablation study (section IV-D).

A. Experimental Setting

Datasets. We evaluate on three datasets covering various sensing modalities and scales:



(a) S3DIS (b) KITTI360 (c) DALES (d) Computational Effiency

Fig. 5: **Oversegmentation Performance.** Oracle mIoU as a function of the number of superpoints on S3DIS, KITTI-360, and DALES. We also report the throughput (from raw points to superpoints) on S3DIS, with error bars indicating variance across configurations. EZ-SP achieves partition purity comparable to or better than PCP while being *over 13× faster*.

- **S3DIS** [47]: Indoor scans of six large building floors, totaling 273M points annotated in 13 classes. Following [15], we use the *merged* version, where each floor is treated as a single point cloud.
- **KITTI-360** [48]: Mobile mapping LiDAR with 919M points and 15 classes, spanning 300 large-scale urban scenes, including 61 validation scans.
- **DALES** [49]: Aerial LiDAR over urban and suburban areas with 492M points across 8 classes, comprising 40 scans, 12 reserved for evaluation.

We subsample all point clouds on a regular grid: 3 cm for S3DIS, 10 cm for KITTI-360 and DALES.

Implementation Details. For partitioning, we fix τ=1 in eq. (1), $w_{p,q}$=1, and build adjacency from the 8-nearest neighbors. The regularization in eq. (3) is λ=0.02, and the intra-edge sampling ratio in eq. (2) is ρ_{intra}=0.1 for S3DIS and 0.3 for KITTI-360/DALES.

The backbone ϕ^{point} is a sparse CNN [16] implemented with TorchSparse [50], with three layers of width [32, 32, 32]. Kernels are 3^3 except in the first layer for KITTI-360/DALES (7^3). The embedding dimension is M=32, yielding models of 58k (S3DIS), 89k (KITTI-360), and 67k (DALES) parameters. We compute three-level hierarchical partitions with minimal superpoint sizes of [5, 30, 90] (S3DIS/KITTI-360) and [5, 15, 70] (DALES).

For segmentation, we use a modified SPT-64 on S3DIS and DALES, and an SPT-128 on KITTI-360: we add a third hierarchical stage, reinstate the feed-forward layer in the transformer block (for DALES only), and concatenate color, position, and elevation with the CNN feature map to form the point embeddings. This yields segmentation heads with 330k (S3DIS), 870k (KITTI-360), and 425k (DALES) parameters. To fight class imbalance, we trained with a focal loss [51] (γ = 1 for S3DIS and γ = 2 for KITTI-360/DALES). All models are trained with Adam (default hyperparameters) and cosine learning rate scheduling with 50 warm-up epochs.

B. Oversegmentation Results

We compare the quality of EZ-SP's partitions against classical and learning-based oversegmentation methods.

Metrics. We follow the common practice of evaluating *oversegmentation*, *i.e.* partitioning a point cloud into compact regions that ideally align with semantic objects. Since the downstream classifier operates on superpoints, partition quality is measured by two criteria: the *number of superpoints* produced and the *oracle mIoU* [45], defined as the mIoU obtained by assigning each superpoint its majority ground-truth label. This provides an upper bound on the segmentation accuracy achievable with a given partition. We also report throughput, obtained either from official model logs or from our own re-runs. All measurements are conducted on comparable Ampere- or Ada-generation GPUs.

Baselines and Competing Methods. We compare against:

- **Voxelization:** Uniform voxel grid grouping.
- **VCCS** [8]: A classical voxel-based oversegmentation based on k-means.
- **Parallel Cut Pursuit (PCP)** [11]: The updated graph-cut partitioner [12] used in SPT [10].
- **SPNet** [9]: Learns partitions via differentiable k-means.

Results. Figure 5 reports the purity of the partition obtained by all methods on three benchmarks, along with their throughput. EZ-SP consistently achieves a purity higher or on par with the best oversegmentation methods for the same number of superpoints, while being a full *order of magnitude faster*. Although our greedy solver yields an objective value roughly 25% higher than PCP when minimizing eq. (3), the resulting partitions exhibit comparable semantic purity in practice. The purity of VCCS's and SPNet's superpoints is limited by their reliance on the rigid k-means algorithm. Moreover, VCCS's CPU-based implementation limits its throughput. SPNet, despite being trained for partitioning, underperforms in purity and requires ~6 h of training, against fewer than 20 minutes for EZ-SP. Voxelization naturally remains the fastest partitioning algorithm, but also yields the least semantically pure partitions.

Qualitative Analysis. We report qualitative examples of partitions in fig. 4. The k-means–based method VCCS fails to adapt to local complexity and produces partitions that

TABLE I: **Efficiency and Performance.** End-to-end processing time on the full S3DIS dataset (273M points), broken down by stage. We also compare model size and semantic segmentation performance on S3DIS, KITTI-360, and DALES.
: preprocessing : partition : semantic segmentation : total time

	Model	Inference time (in GPU-s) ↓				Size ↓	Performance (mIoU) ↑			
							S3DIS		K-360	DALES
						$\times 10^6$ params.	6-Fold	Area 5	val	test
point/voxel	PointNet++ [52]	125	-	52	177	3.0	56.7	-	-	68.3
	KPConv [15]	1031	-	354	1385	14.1	70.6	67.1	-	**81.1**
	MinkowskiNet [14]	887	-	302	1189	37.9	69.1	65.4	58.3	-
	PointNeXt-XL [53]	-	-	66k	66k	41.6	74.9	71.1	-	-
	Strat. Trans. [54]	-	-	26k	26k	8.0	74.9	72.0	-	74.3
	PTv3 [13]	-	-	11k	11k	46.2	**77.7**	**73.4**	-	-
superpoint	SPG [7]	3187	2616	56	5859	0.28	62.1	58.0	-	60.6
	SSP [45]	3220	2616	56	5892	0.29	68.4	61.7	-	-
	SPNet [9]	3187	445	56	3688	0.33	68.7	-	-	-
	SPT [10]	376	418	**14**	808	**0.21**	76.0	68.9	**63.5**	79.6
	EZ-SP(ours)	**136**	**3**	**14**	**153**	0.39	76.1	69.6	62.0	79.4

resemble uniform tessellations. PCP adapts better to variations in complexity, but still tessellates large, simple surfaces. In contrast, EZ-SP yields the most adaptive partitions: it produces large, semantically pure superpoints on simple structures such as ground, roofs, or walls, while allocating small superpoints to geometrically complex regions.

C. Semantic Segmentation

Table I reports both accuracy and efficiency for a range of SOTA semantic segmentation models. All methods are trained only on the official training split of each dataset, without using any external data.

Analysis. Superpoint-based methods consistently deliver competitive accuracy with *100–200*× fewer parameters than typical point- or voxel-based networks. Yet their inference speed is often constrained by the CPU-bound partitioning stage, which can dominate runtime. EZ-SP removes this bottleneck, yielding much faster end-to-end inference while retaining top-tier accuracy. With fewer than 400k parameters (under 2 MB of VRAM), it ranks among the smallest models in this comparison—orders of magnitude lighter than many voxel- or point-based alternatives. In terms of inference speed, EZ-SP is unmatched: faster even than PointNet++, while maintaining strong performance across all datasets. Its accuracy trails SOTA large models by less than 2 mIoU points, comparable to SPT and well within the expected variance of training and macro-averaged evaluation metrics.

In contrast, modern point- and voxel-based architectures such as PointNeXt [53], Stratified Transformer [54], and PointTransformer-v3 [13] can be extremely slow at inference, in part due to heavy test-time augmentation (TTA). Removing TTA reduces accuracy by −1.5 mIoU on S3DIS Area 5 for Stratified Transformer and by −1.6 for PTv3, yet they still remain significantly slower than our approach (e.g., PTv3 requires 988 s on S3DIS without TTA).

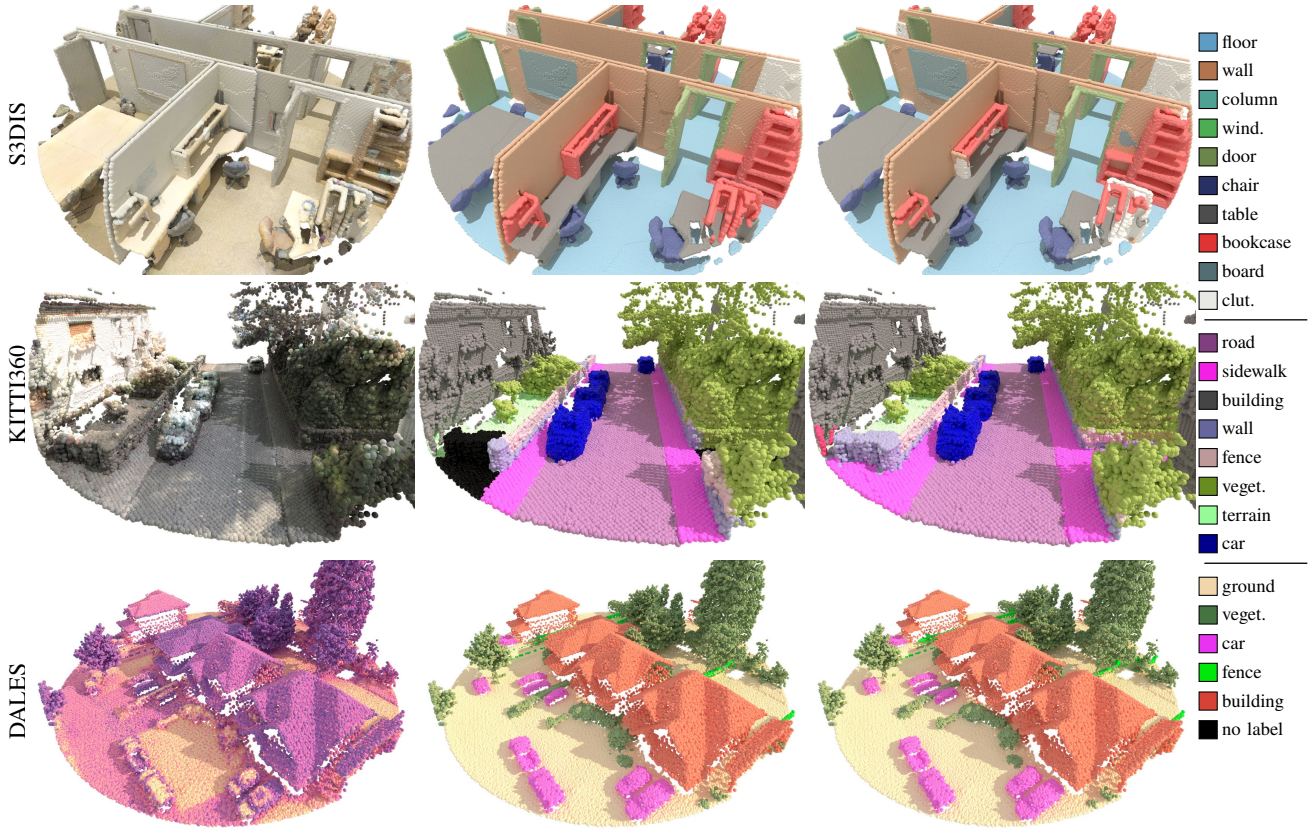
Qualitative Analysis. Figure 6 shows semantic segmentations produced by EZ-SP across indoor, outdoor, and aerial domains. Predictions remain reliable even in complex environments, with most errors arising from the ambiguous `clutter` class, and cases that are inherently difficult to partition into superpoints due to low geometric and radiometric contrasts at their interface, such as a whiteboard against a white wall. At the same time, EZ-SP accurately captures fine details such as furniture edges, vehicle boundaries, and roof structures. Overall, it achieves high-quality segmentation while delivering orders of magnitude faster inference than existing approaches.

TABLE II: **Scalability of EZ-SP.** From a single LiDAR scan to a city-scale aerial survey, entire scenes can be processed in one pass on embedded or commercial-grade GPU memory.

Scenario	Points	VRAM	Compatible Hardware
Autonomous driving single LiDAR scan	105k pts	EZ-SP: 0.25 GB MinkowskiNet: 0.52 GB	Jetson-Nano
Digital twin building-scale	79M pts	EZ-SP: 29 GB MinkowskiNet: 30 GB	2× Radeon RX 7900: 40GB
Aerial survey city-scale, 1.3 km^2	16M pts	EZ-SP: 45 GB MinkowskiNet : OoM	A40 48GB

Breaking the Partition Bottleneck. Figure 7 details the breakdown of end-to-end inference time across different pipeline stages for Superpoint Transformer [10] and our proposed EZ-SP—excluding I/O, which largely depends on dataset format, *e.g.* plain `.txt` vs. binary `.ply` files. For SPT, the CPU-bound partition stage dominates computation time; in EZ-SP, this cost is virtually eliminated due to our GPU-based implementation and the removal of handcrafted point features. Additional minor optimizations further reduce runtime in other stages. Beyond speed, EZ-SP is also easier to deploy and tune, as it avoids the complex feature engineering and hyperparameters optimization required by traditional superpoint partitioning methods.

Memory Efficiency. As shown in table II, EZ-SP processes full scans *in a single pass, without tiling*, across scenarios ranging from real-time inference on a single Velodyne64 sweep to city-scale aerial surveys. A single LiDAR rotation runs on an embedded Jetson device (< $200), while an entire S3DIS floor (68 rooms) fits on a consumer GPU (< $1000). Even 1.3 km^2 of aerial LiDAR from DALES can be handled at once on an NVIDIA A40 (~ $3000).



(a) Input (b) Ground Truth (c) Prediction

Fig. 6: **Semantic Segmentation.** Qualitative results on three benchmarks: S3DIS, KITTI-360, and DALES. For each dataset, we show the input point cloud (RGB for S3DIS and KITTI-360, LiDAR intensity for DALES), the ground-truth labels, and the predictions of EZ-SP.

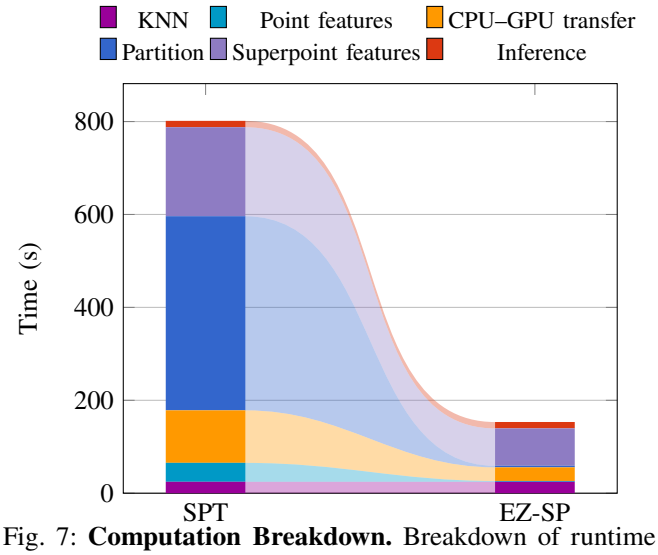


Fig. 7: **Computation Breakdown.** Breakdown of runtime comparison between SPT and EZ-SP for S3DIS 6-Fold.

D. Ablation Study

We assess the contribution of each component of our approach through the ablations summarized in table III. For ablations A, B, and D, we retrain our SPT model from scratch.

- A: **Learning to Partition.** Replacing our lightweight CNN (section IV-B) with handcrafted geometric features [55], [31] yields comparable performance. This demonstrates that our approach eliminates the need for handcrafted inputs, as the CNN learns features of similar expressivity, while reducing manual engineering.
- **B: Hierarchical Levels.** Adding a third hierarchical level to our SPT improves accuracy with negligible impact on throughput.
- C: **Optimizations.** The various implementation optimizations over the original SPT codebase (CPU-GPU transfer, GPU-bound feature computation) do not impact the performance but double our throughput.
- **D: Partition Algorithm.** Replacing our GPU partition algorithm with the original CPU-based PCP [11] based on handcrafted features (as in SPT) reduces throughput substantially, while delivering slightly lower accuracy. As in Robert et al. [10], we find that the 3-level PCP partition does not improve performance, unlike ablation B, suggesting that despite comparable oracle mIoU, our superpoint partition's hierarchical structure is more informative than PCP's.

V. Conclusion

We presented EZ-SP, a fast and lightweight superpoint partitioning model that removes the long-standing partitioning bottleneck in superpoint-based 3D semantic segmentation.

TABLE III: **Ablation.** Performance of variants of EZ-SP.

	Configuration	S3DIS Fold5 mIoU	Throughput $\times 10^6$ pt/s
	Best	69.6	1.7
A	Handcrafted features	69.5	1.7
B	2 hierarchical levels	67.2	1.7
C	No optimization	-	0.8
D	PCP	67.8	0.3

Our end-to-end pipeline can process 1.7M points/s on a single consumer grade GPU and achieves near–state-of-the-art accuracy in indoor, terrestrial, and aerial LiDAR benchmarks. By drastically lowering runtime costs, EZ-SP opens the door to efficient analysis of massive 3D scans and real-time perception on resource-constrained and embedded platforms.

References

[1] R. Loiseau, M. Aubry, and L. Landrieu, "Online segmentation of LiDAR sequences: Dataset and algorithm," *ECCV*, 2022.

[2] S. Xu, D. Honegger, M. Pollefeys, and L. Heng, "Real-time 3D navigation for autonomous vision-guided MAVs," in *IROS*, 2015.

[3] F. L. Busch, T. Homberger, J. Ortega-Peimbert, Q. Yang, and O. Andersson, "One map to find them all: Real-time open-vocabulary mapping for zero-shot multi-object navigation," in *ICRA*, 2025.

[4] P. Pfaff, R. Triebel, C. Stachniss, P. Lamon, W. Burgard, and R. Siegwart, "Towards mapping of cities," in *ICRA*, 2007.

[5] Z. Makhataeva and H. A. Varol, "Augmented reality for robotics: A review," *Robotics*, 2020.

[6] N. K. Beigi, B. Partov, and S. Farokhi, "Real-time cloud robotics in practical smart city applications," in *PIMRC*, 2017.

[7] L. Landrieu and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," *CVPR*, 2018.

[8] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter, "Voxel cloud connectivity segmentation-supervoxels for point clouds," *CVPR*, 2013.

[9] L. Hui, J. Yuan, M. Cheng, J. Xie, X. Zhang, and J. Yang, "Superpoint network for point cloud oversegmentation," *ICCV*, 2021.

[10] D. Robert, H. Raguet, and L. Landrieu, "Efficient 3D semantic segmentation with superpoint transformer," in *ICCV*, 2023.

[11] H. Raguet and L. Landrieu, "Parallel cut pursuit for minimization of the graph total variation," *ICML Workshop on Graph Reasoning*, 2019.

[12] L. Landrieu and G. Obozinski, "Cut pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs," in *SIAM Journal on Imaging Sciences*, 2017.

[13] X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, and H. Zhao, "Point transformer v3: Simpler faster stronger," in *CVPR*, 2024.

[14] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal ConvNets: Minkowski convolutional neural networks," *CVPR*, 2019.

[15] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "KPConv: Flexible and deformable convolution for point clouds," *ICCV*, 2019.

[16] B. Graham, M. Engelcke, and L. Van Der Maaten, "3D semantic segmentation with submanifold sparse convolutional networks," *CVPR*, 2018.

[17] L. Hui, H. Yang, M. Cheng, J. Xie, and J. Yang, "Pyramid point cloud transformer for large-scale place recognition," in *ICCV*, 2021.

[18] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "Pct: Point cloud transformer," *CVM*, 2021.

[19] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," *ICCV*, 2021.

[20] X. Wu, Y. Lao, L. Jiang, X. Liu, and H. Zhao, "Point transformer v2: Grouped vector attention and partition-based pooling," in *NeurIPS*, 2022.

[21] Z. Liang, Z. Li, S. Xu, M. Tan, and K. Jia, "Instance segmentation in 3D scenes using semantic superpoint tree networks," *CVPR*, 2021.

[22] J. Sun, C. Qing, J. Tan, and X. Xu, "Superpoint transformer for 3D scene instance segmentation," in *AAAI*, 2023.

[23] D. Robert, H. Raguet, and L. Landrieu, "Scalable 3D panoptic segmentation as superpoint graph clustering," in *3DV*, 2024.

[24] Z. Liu, X. Qi, and C.-W. Fu, "One thing one click: A self-training approach for weakly supervised 3D semantic segmentation," in *CVPR*, 2021.

[25] Z. Zhang, B. Yang, B. Wang, and B. Li, "GrowSP: Unsupervised semantic segmentation of 3D point clouds," *CVPR*, 2023.

[26] L. Nunes, R. Marcuzzi, X. Chen, J. Behley, and C. Stachniss, "SegContrast: 3D point cloud feature representation learning through self-supervised segment discrimination," *IEEE Robotics and Automation Letters*, 2022.

[27] J. Liu, Z. Yu, T. P. Breckon, and H. P. H. Shum, "U3DS3: Unsupervised 3D semantic scene segmentation," *WACV*, 2023.

[28] G. Gao, M. Lauri, J. Zhang, and S. Frintrop, "Saliency-guided adaptive seeding for supervoxel segmentation," in *IROS*, 2017.

[29] Y. Lin, C. Wang, D. Zhai, W. Li, and J. Li, "Toward better boundary preserved supervoxel segmentation for 3D point clouds," *ISPRS journal of photogrammetry and remote sensing*, 2018.

[30] Y. Ben-Shabat, T. Avraham, M. Lindenbaum, and A. Fischer, "Graph based over-segmentation methods for 3D point clouds," *Computer Vision and Image Understanding*, 2017.

[31] S. Guinard and L. Landrieu, "Weakly supervised segmentation-aided classification of urban scenes from 3D LiDAR point clouds," *ISPRS Workshop*, 2017.

[32] L. Han, T. Zheng, L. Xu, and L. Fang, "Occuseg: Occupancy-aware 3D instance segmentation," *CVPR*, 2020.

[33] A. Thyagharajan, B. Ummenhofer, P. Laddha, O. J. Omer, and S. Subramoney, "Segment-fusion: Hierarchical context fusion for robust 3D semantic segmentation," *CVPR*, 2022.

[34] S. Jin, I. Armeni, M. Pollefeys, and D. Barath, "Multiway point cloud mosaicking with diffusion and global optimization," in *CVPR*, 2024.

[35] Z. J. Yew and G. H. Lee, "Regtr: End-to-end point cloud correspondences with transformers," in *CVPR*, 2022.

[36] Y. Zhu, L. Hui, Y. Shen, and J. Xie, "SPGroup3D: Superpoint grouping network for indoor 3D object detection," in *AAAI*, 2024.

[37] Y. Yang, X. Wu, T. He, H. Zhao, and X. Liu, "SAM3D: Segment anything in 3D scenes," *arXiv:2306.03908*, 2023.

[38] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *TPAMI*, 2012.

[39] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *ICCV*, 2023.

[40] M. Cheng, L. Hui, J. Xie, J. Yang, and H. Kong, "Cascaded non-local neural network for point cloud semantic segmentation," in *ROS*, 2020.

[41] D. Lu, L. Xu, J. Zhou, K. Y. Gao, and J. Li, "3DLST: 3D learnable supertoken transformer for LiDAR point cloud scene segmentation," *JAG*, 2025.

[42] X. Kang, C. Wang, and X. Chen, "Region-enhanced feature learning for scene semantic segmentation," *arXiv:2304.07486*, 2023.

[43] L. Hui, L. Tang, Y. Shen, J. Xie, and J. Yang, "Learning superpoint graph cut for 3D instance segmentation," *NeurIPS*, 2022.

[44] H. Guo, H. Zhu, S. Peng, Y. Wang, Y. Shen, R. Hu, and X. Zhou, "SAM-guided graph cut for 3D instance segmentation," in *ECCV*, 2024.

[45] L. Landrieu and M. Boussaha, "Point cloud oversegmentation with graph-structured deep metric learning," *CVPR*, 2019.

[46] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop*, 2019.

[47] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3D semantic parsing of large-scale indoor spaces," *CVPR*, 2016.

[48] Y. Liao, J. Xie, and A. Geiger, "KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2D and 3D," *TPAMI*, 2022.

[49] N. Varney, V. K. Asari, and Q. Graehling, "DALES: A large-scale aerial LiDAR data set for semantic segmentation," *CVPRW*, 2020.

[50] H. Tang, S. Yang, Z. Liu, K. Hong, Z. Yu, X. Li, G. Dai, Y. Wang, and S. Han, "Torchsparse++: Efficient training and inference framework for sparse convolution on gpus," in *MICRO*, 2023.

[51] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *ICCV*, 2017.

[52] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," *NeurIPS*, 2017.

[53] G. Qian, Y. Li, H. Peng, J. Mai, H. Hammoud, M. Elhoseiny, and B. Ghanem, "PointNeXt: Revisiting PoinNet++ with improved training and scaling strategies," *NeurIPS*, 2022.

[54] X. Lai, J. Liu, L. Jiang, L. Wang, H. Zhao, S. Liu, X. Qi, and J. Jia, "Stratified transformer for 3D point cloud segmentation," *CVPR*, 2022.

[55] J. Demantké, C. Mallet, N. David, and B. Vallet, "Dimensionality based scale selection in 3D LiDAR point clouds," in *Laserscanning*, 2011.