Efficient 3D Semantic Segmentation with Superpoint Transformer

Damien Robert, Hugo Raguet, Loic Landrieu

Efficient 3D Semantic Segmentation with Superpoint Transformer

Damien Robert[1, 2] damien.robert@ign.fr
Hugo Raguet[3] hugo.raguet@insa-cvl.fr
Loic Landrieu[2,4] loic.landrieu@enpc.fr

[1]CSAI, ENGIE Lab CRIGEN, France
[2] LASTIG, IGN, ENSG, Univ Gustave Eiffel, France
[3]INSA Centre Val-de-Loire Univ de Tours, LIFAT, France
[4]LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, France

Abstract

We introduce a novel superpoint-based transformer architecture for efficient semantic segmentation of large-scale 3D scenes. Our method incorporates a fast algorithm to partition point clouds into a hierarchical superpoint structure, which makes our preprocessing 7 times times faster than existing superpoint-based approaches. Additionally, we leverage a self-attention mechanism to capture the relationships between superpoints at multiple scales, leading to state-of-the-art performance on three challenging benchmark datasets: S3DIS (76.0% mIoU 6-fold validation), KITTI-360 (63.5% on Val), and DALES (79.6%). With only 212k parameters, our approach is up to 200 times more compact than other state-of-the-art models while maintaining similar performance. Furthermore, our model can be trained on a single GPU in 3 hours for a fold of the S3DIS dataset, which is 7× to 70× fewer GPU-hours than the best-performing methods. Our code and models are accessible at `github.com/drprojects/superpoint_transformer`.

1. Introduction

As the expressivity of deep learning models rapidly increases, so do their complexity and resource requirements [15]. In particular, vision transformers have demonstrated remarkable results for 3D point cloud semantic segmentation [62, 42, 18, 26, 37], but their high computational requirements make them challenging to train effectively. Additionally, these models rely on regular grids or point samplings, which do not adapt to the varying complexity of 3D data: the same computational effort is allocated everywhere, regardless of the local geometry or radiometry of the point cloud. This issue leads to needlessly high memory consumption, limits the number of points that can be processed simultaneously, and hinders the modeling of long-range interactions.

Superpoint-based methods [30, 27, 23, 46] address the limitation of regular grids by partitioning large point clouds into sets of points— superpoints—which adapt to the local complexity. By directly learning the interaction between superpoints instead of individual points, these methods enable the analysis of large scenes with compact and parsimonious models that can be trained faster than standard approaches. However, superpoint-based methods often require a costly preprocessing step, and their range and expressivity are lim-

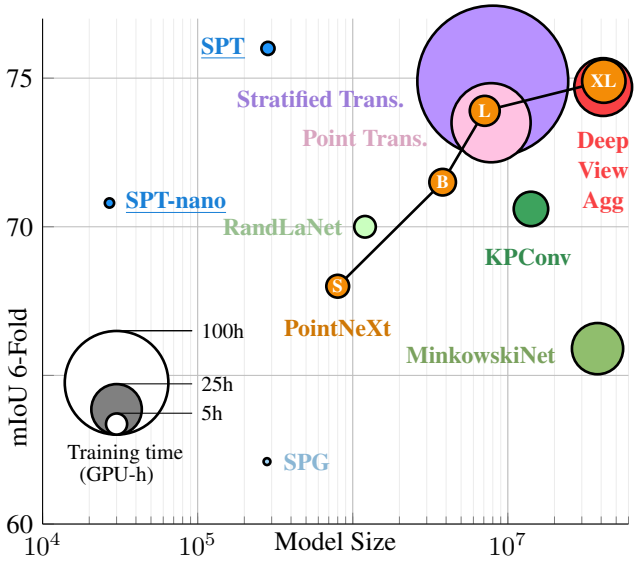


Figure 1: **Model Size vs. Performance.** We visualize the performance of different methods on the S3DIS dataset (6-fold validation) in relation to their model size in log-scale. The area of the markers indicates the GPU-time to train on a single fold. Our proposed method Superpoint Transformer (SPT) achieves state-of-the-art with a reduction of up to 200-fold in model size and 70-fold in training time (in GPU-h) compared to recent methods. The even smaller SPT-nano model achieves a fair performance with 26k parameters only.

ited by their use of local graph-convolution schemes [52].

In this paper, we propose a novel superpoint-based transformer architecture that overcomes the limitations of both approaches, see Figure 1. Our method starts by partitioning a 3D point cloud into a hierarchical superpoint structure, which adapts to the local properties of the acquisition at multiple scales simultaneously. To compute this partition efficiently, we propose a new algorithm that is an order of magnitude faster than existing superpoint preprocessing algorithms. Next, we introduce the Superpoint Transformer (SPT) architecture, which uses a sparse self-attention scheme to learn relationships between superpoints at multiple scales. By viewing the semantic segmentation of large point clouds as the classification of a small number of superpoints, our model can accurately classify millions of 3D points simultaneously without relying on sliding windows. SPT achieves near state-of-the-art accuracy on various open benchmarks while being significantly more compact and able to train much quicker than common approchaes. The main contributions of this paper are as follows:

- **Efficient Superpoint Computation:** We propose a new method to compute a hierarchical superpoint structure for large point clouds, which is more than 7 times faster than existing superpoint-based methods. Our preprocessing time is also comparable or faster than standard approaches, addressing a significant drawback of superpoint methods.
- **State-of-the-Art Performance:** Our model reaches performance at or close to the state-of-the-art for three open benchmarks with distinct settings: S3DIS for indoor scanning [3], KITTI-360 for mobile outdoor acquisitions [33], and DALES for city-scale aerial LiDAR [56].
- **Resource-Efficient Models:** SPT is particularly resource-efficient as it only has 212k parameters for S3DIS and DALES, a 200-fold reduction compared to other state-of-the-art models such as PointNeXt [45] and takes 70 times fewer GPU-h to train than Stratified Transformer [26]. The even more compact SPT-nano reaches 70.8% 6-Fold mIoU on S3DIS with only 26k parameters, making it the smallest model to reach above 70% by a factor of almost 300.

2. Related Work

This section provides an overview of the main inspirations for this paper, which include 3D vision transformers, partition-based methods, and efficient learning for 3D data.

3D Vision Transformers. Following their adoption for image processing [10, 35], Transformer architectures [57] designed explicitly for 3D analysis have shown promising results in terms of performance [62, 18] and speed [42, 37]. In particular, the Stratified Transformer of Lai *et al*. uses a specific sampling scheme [26] enabling to model long-range interactions. However, the reliance of 3D vision transformers on arbitrary K-nearest or voxel neighborhoods leads to high memory consumption which hinders the processing of large scenes and the ability to leverage global context cues.

Partition-Based Methods. Partitioning images into superpixels has been studied extensively for simplifying image analysis, both before and after the widespread use of deep learning [1, 55]. Similarly, superpoints are used for 3D point cloud segmentation [41, 34] and object detection [19, 11]. SuperPointGraph [30] proposed to learn the relationship between superpoints using graph-convolutions [52] for semantic segmentation. While this method trains fast, its preprocessing is slow and its expressivity and range are limited as it operates on a single partition. Recent works have proposed ways of learning the superpoints themselves [27, 23, 54], which yields improved results but at the cost of an extra training step, or a large point-based backbone [25].

Hierarchical partitions are used for image processing [2, 60, 61] and 3D analysis tasks such as point cloud compression [12] and object detection [7, 32]. Hierarchical approaches for semantic segmentation use Octrees with fixed grids [40, 49]. In contrast, SPT uses a multi-scale hierarchical structure that adapts to the local geometry of the data. This leads to partitions which conform more closely to semantic boundaries, enabling the network to model the interactions between objects or object parts.

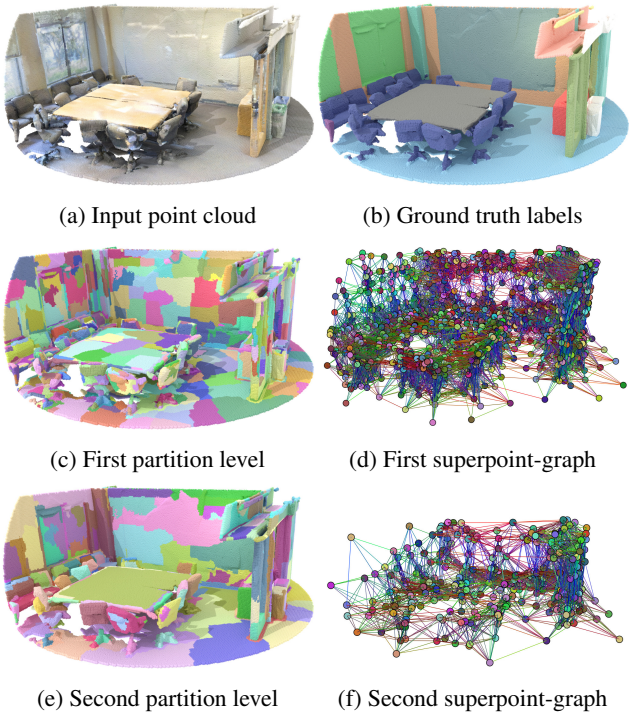
Efficient 3D Learning. As 3D scans of real-world scenes can contain hundreds of millions of points, optimizing the efficiency of 3D analysis is an essential area of research. PointNeXt [45] proposes several effective techniques that enable simple and efficient methods [44] to achieve state-of-the-art performance. RandLANet [22] demonstrates that efficient sampling strategies can yield excellent results. Sparse [16] or hybrid [36] point cloud representations have also helped reduce memory usage. However, by leveraging the local similarity of dense point clouds, superpoint-based methods can achieve an input reduction of several orders of magnitude, resulting in unparalleled efficiency.

3. Method

Our method has two key components. First, we use an efficient algorithm to segment an input point cloud into a compact multi-scale hierarchical structure. Second, a transformer-based network leverages this structure to classify the elements of the finest scale.

3.1. Efficient Hierarchical Superpoint Partition

We consider a point cloud $\mathcal{C}$ with positional and radiometric information. In order to learn multi-scale interactions, we compute a hierarchical partition of $\mathcal{C}$ into geometrically-homogeneous superpoints of increasing coarseness, see Figure 2. We first define the concept of hierarchical partitions.



(a) Input point cloud (b) Ground truth labels

(c) First partition level (d) First superpoint-graph

(e) Second partition level (f) Second superpoint-graph

Figure 2: **Superpoint Transformer.** Our method takes as input a point cloud a) and computes its hierarchical partition into geometrically homogeneous superpoints at multiple scales: c) and e). For all partition levels, we construct superpoint adjacency graphs d) and f), which are used by an attention-based network to classify the finest superpoints.

Definition 1 Hierarchical Partitions. A partition of a set $\mathcal{X}$ is a collection of subsets of $\mathcal{X}$ such that each element of $\mathcal{X}$ is in one and only one such subset. $\mathcal{P} := [\mathcal{P}_0, \cdots, \mathcal{P}_I]$ is a hierarchical partition of $\mathcal{X}$ if $\mathcal{P}_0 = \mathcal{X}$, and $\mathcal{P}_{i+1}$ is a partition of $\mathcal{P}_i$ for $i \in [0, I-1]$.

Hierarchical Superpoint Partitions. We propose an efficient approach for constructing hierarchical partitions of large point clouds. We first associate each point c of $\mathcal{C}$ with a feature f_c representing its local geometric and radiometric information. These features can be handcrafted [17] or learned [27, 23]. See the Appendix for more details on point features. We also define a graph $\mathcal{G}$ encoding the adjacency between points usually based on spatial proximity, *e.g.* k-nearest neighbours.

We view the features f_c for all c of $\mathcal{C}$ as a signal f defined on the nodes of the graph $\mathcal{G}$. Following the ideas of SuperPoint Graph [30], we compute an approximation of f into constant components by solving an energy minimization problem penalized with a graph-based notion of *simplicity*. The resulting constant components form a partition whose granularity is determined by a regularization strength $\lambda > 0$: higher values yield fewer, coarser components.

For each component of the partition, we can compute the mean position (centroid) and feature of its elements, defining a coarser point cloud on which we can repeat the partitioning process. We can now compute a hierarchical partition $\mathcal{P} := [\mathcal{P}_0, \cdots, \mathcal{P}_I]$ of $\mathcal{C}$ from a list of increasing regularization strengths $\lambda_1, \cdots, \lambda_I$. First, we set $\mathcal{P}_0$ as the point cloud $\mathcal{C}$ and f^0 as the point features f. Then, for $i = 1$ to I, we compute (i) a partition $\mathcal{P}_i$ of f^{i-1} penalized with λ_i; (ii) the mean signal f^i for all components of $\mathcal{P}_i$. The coarseness of the resulting partitions $[\mathcal{P}_0, \cdots, \mathcal{P}_I]$ is thus strictly increasing. See the Appendix for a more detailed description of this process.

Hierarchical Partition Structure. A hierarchical partition defines a polytree structure across the different levels. Let p be an element of $\mathcal{P}_i$. If $i \in [0, I-1]$, $\text{parent}(p)$ is the component of $\mathcal{P}_{i+1}$ which contains p. If $i \in [1, I]$, $\text{children}(p)$ is the set of component of $\mathcal{P}_{i-1}$ whose parent is p.

Superpoints also share adjacency relationships with superpoints *of the same partition level*. For each level $i \geq 1$, we build a *superpoint-graph* $\mathcal{G}_i$ by connecting adjacent components of $\mathcal{P}_i$, *i.e.* superpoints whose closest points are within a gap distance $\epsilon_i > 0$. The list of superpoint-graphs $\mathcal{G}_1, \cdots, \mathcal{G}_I$ allows us to define $\mathcal{N}(p) \subset \mathcal{P}_i$, the neighborhood of p within its partition level i. More details on the superpoint-graph construction can be found in the Appendix.

Hierarchical Parallel ℓ_0-Cut Pursuit. Computing the hierarchical components involves solving a recursive sequence of non-convex, non-differentiable optimization problems on large graphs. We propose an adaptation of the ℓ_0-cut pursuit algorithm [29] to solve this problem. To improve efficiency, we adapt the graph-cut parallelization strategy initially introduced by Raguet *et al*. [47] in the convex setting.

3.2. Superpoint Transformer

Our proposed SPT architecture draws inspiration from the popular U-Net [51, 14]. However, instead of using grid, point, or graph subsampling, our approach derives its different resolution levels from the hierarchical partition $\mathcal{P}$.

General Architecture. As represented in Figure 3, SPT comprises an encoder with $I-1$ stages and a decoder with $I-2$ stages: the prediction takes place at level $\mathcal{P}_1$ and not individual points. We start by computing the relative positions x of all points and superpoints with respect to their parent. For a superpoint $p \in \mathcal{P}_i$, we define x_p^i as the position of the centroid of p relative to its parent's. The coarsest superpoints of $\mathcal{P}_I$ have no parent and use the center of the scene as a reference centroid. We then normalize these values such that the sets $\{x_p^i | p \in \text{children}(q)\}$ have a radius of 1 for all $q \in \mathcal{P}_{i+1}$. We compute features for each 3D point

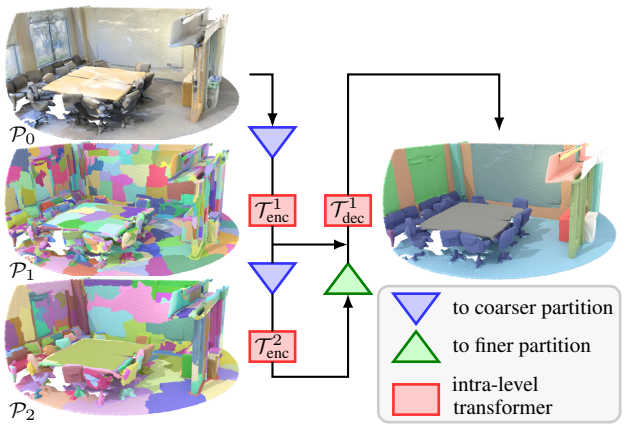


Figure 3: **Superpoint Transformer.** We represent our proposed architecture with two partitions levels $\mathcal{P}_1$ and $\mathcal{P}_2$. We use a transformer-based module to leverage the context at different scales, leading to large receptive fields. We only classify the superpoints of the partition $\mathcal{P}_1$ and not individual 3D points, allowing fast training and inference.

by using a multi-layer perceptron (MLP) to mix their relative positions and handcrafted features: $g^0 := \phi^0_{\text{enc}}([x^0, f^0])$, with $[\cdot,\cdot]$ the channelwise concatenation operator.

Each level $i > 1$ of the encoder maxpools features from the finer partition level $i-1$, adds relative positions x^i and propagates information between neighboring superpoints in $\mathcal{G}_i$. For a superpoint p in $\mathcal{P}_i$, this translates as:

$$g^i_p = \mathcal{T}^i_{\text{enc}} \circ \phi^i_{\text{enc}} \left(\left[x^i_p, \max_{q \in \text{children}(p)} \left(g^{i-1}_q \right) \right] \right) \quad (1)$$

with ϕ^i_{enc} an MLP and $\mathcal{T}^i_{\text{enc}}$ a transformer module explained below. By avoiding communication between 3D points, we bypass a potential computational bottleneck.

The decoder passes information from the coarser partition level $i+1$ to the finer level i. It uses the relative positions x^i and the encoder features g^i to improve the spatial resolution of its feature maps h^i [51]. For a superpoint p in partition $\mathcal{P}_i$ with $1 < i < I-1$, this can be expressed as:

$$h^i_p = \mathcal{T}^i_{\text{dec}} \circ \phi^i_{\text{dec}} \left(\left[x^i_p, g^i_p, h^{i+1}_{\text{parent}(p)} \right] \right) \quad (2)$$

with $h^I = g^I$, ϕ^i_{dec} an MLP, and $\mathcal{T}^i_{\text{dec}}$ an attention-based module similar to $\mathcal{T}^i_{\text{enc}}$.

Self-Attention Between Superpoints. We propose a variation of graph-attention networks [58] to propagate information between neighboring superpoints of the same partition level. We associate to each superpoint $p \in \mathcal{P}_i$ a key, query, value triplet K_p, Q_p, V_p of respective size $D_{\text{key}}, D_{\text{key}}$ and D_{val}. These vectors are obtained by applying a linear layer to the feature map g^i normalized with GraphNorm [5].

We characterize the adjacency relationship between p and its M neighbors $\mathcal{N}(p)$ in $\mathcal{G}_i$ with the triplet $A^{\text{key}}_p, A^{\text{que}}_p, A^{\text{val}}_p$ of respective dimension $M \times D_{\text{key}}, M \times D_{\text{key}}$ and $M \times D_{\text{val}}$. The computation of these adjacency features is described in the next section. The modules $\mathcal{T}^i_{\text{enc}}$ and $\mathcal{T}^i_{\text{dec}}$ gather contextual information from neighbouring superpoints as follows:

$$[\mathcal{T}(f)]_p \overset{+}{=} \text{att}(Q_p \oplus A^{\text{que}}_p, K_{\mathcal{N}(p)} + A^{\text{key}}_p, V_{\mathcal{N}(p)} + A^{\text{val}}_p) \ , \quad (3)$$

with $\overset{+}{=}$ a residual connection [20] and $\oplus$ the addition operator with broadcasting on dimension M. The attention mechanism writes:

$$\text{att}(Q, K, V) := V^\intercal \, \text{softmax} \left(\frac{Q \odot K\mathbf{1}}{\sqrt{M}} \right) \ , \quad (4)$$

with $\odot$ the Hadamard termwise product and $\mathbf{1}$ a column-vector with D_{key} ones. Our proposed scheme is similar to classic attention schemes with two differences: (i) the queries adapt to each neighbors, and (ii) we normalize the softmax with the neighbourhood size instead of the key dimension. In practice, we use multiple independent attention modules in parallel (multi-head attention) and several consecutive attention blocks.

3.3. Leveraging the Hierarchical Partition

The hierarchical superpoint partition $\mathcal{P}$ can be used for more than guidance for graph pooling operations. Indeed, we can learn expressive adjacency encodings capturing the complex adjacency relationships between superpoints, and employ powerful supervision and augmentation strategies based on the hierarchical partitions.

Adjacency Encoding. While the adjacency between 3D points, pixels, and patches is entirely defined by their offset, the relationships between superpoints are governed by additional factors such as their alignment, proximity, or relative size. We characterize the adjacency of pairs of adjacent superpoints of the same partition level using the following handcrafted features: (i) distance between centroids, (ii) distance between closest points, (iii) alignment of normal directions, and (iv) the ratio between the superpoints' length, volume, surface, and point count. These features are efficiently computed only once during the preprocessing.

For each adjacent superpoints in $\mathcal{P}_i$, we map the handcrafted adjacency features to the triplet $A^{\text{key}}, A^{\text{que}}, A^{\text{val}}$ using a dedicated MLP ϕ^i_{adj}. Further details on the superpoint-graph construction and specific adjacency features are provided in the Appendix.

Hierarchical Supervision. We propose to incorporate the nested structure of the hierarchical partition $\mathcal{P}$ into the supervision of our model. We can naturally associate the superpoints of any level $i \geq 1$ with a set of 3D points in $\mathcal{P}_0$. The

superpoints at the finest level $i = 1$ are almost semantically pure (see Figure 6), while the superpoints at coarser levels $i > 1$ typically encompass multiple objects. Therefore, we use a dual learning objective: (i) we predict the most frequent label within the superpoints of $\mathcal{P}_1$ supervised with the categorical cross-entropy, and (ii) we predict the label distribution for the superpoints of $\mathcal{P}_i$ with $i > 1$, supervised with the Kullback–Leibler divergence (KL) [24].

Let z_p^i denote the true label distribution of the 3D points within a superpoint $p \in \mathcal{P}_i$, and $\hat{z}_p^i$ the most frequent label in p. We use a dedicated linear layer at each partition level to map the decoder feature g_p^i to a predicted label distribution y_p^i. Our objective function can be formulated as follows:

$$\mathcal{L} = \sum_{p \in \mathcal{P}_1} \frac{-N_p^1}{|\mathcal{C}|} \log(y_p^1[\hat{z}_p^1]) + \sum_{i=2}^{I} \sum_{p \in \mathcal{P}_i} \frac{\mu^i N_p^i}{|\mathcal{C}|} \mathrm{KL}(z_p^i, y_p^i) \ , \quad (5)$$

where $\mu^2, \cdots, \mu^I$ are positive weights, N_p^i represents the number of points within a superpoint $p \in \mathcal{P}_i$, and $|\mathcal{C}|$ is the total number of points in the point cloud.

Superpoint-Based Augmentations. Although our approach classifies superpoints rather than individual 3D points, we still need to load the points of $\mathcal{P}_0$ in memory to embed the superpoints from $\mathcal{P}_1$. However, since superpoints are designed to be geometrically simple, only a subset of their points is needed to characterize their shape. Therefore, when computing the feature g_p^1 of a superpoint p of $\mathcal{P}_1$ containing n points with Eq. (1), we sample only a portion $\tanh(n/n_{\text{max}})$ of its points, with a minimum of n_{min}. This sampling strategy reduces the memory load and acts as a powerful data augmentation. The lightweight version of our model SPT-nano goes even further. It ignores the points entirely and only use handcrafted features to embed the superpoints of $\mathcal{P}_1$, thus avoiding entirely the complexity associated with the size of the input point cloud $\mathcal{P}_0$.

To further augment the data, we exploit the geometric consistency of superpoints and their hierarchical arrangement. During the batch construction, we randomly drop each superpoint with a given probability at all levels. Dropping superpoints at the fine levels removes random objects or object parts, while dropping superpoints at the coarser levels removes entire structures such as walls, buildings, or portions of roads, for example.

4. Experiments

We evaluate our model on three diverse datasets described in Section 4.1. In Section 4.2, we evaluate our approach in terms of precision, but also quantify the gains in terms of pre-processing, training, and inference times. Finally, we propose an extensive ablation study in Section 4.3.

Table 1: **Partition Configuration.** We report the point count of different datasets before and after subsampling, as well as the size of the partitions.

Dataset	Points	Subsampled	$\lvert \mathcal{P}_1 \rvert$	$\lvert \mathcal{P}_2 \rvert$
S3DIS [3]	273m	32m	979k	292k
DALES [56]	492m	449m	14.8m	2.56m
KITTI-360 [33]	919m	432m	16.2m	2.98m

4.1. Datasets and Models

Datasets. To demonstrate its versatility, we evaluate SPT on three large-scale datasets of different natures.

S3DIS [3]. This indoor dataset of office buildings contains over 274 million points across 6 building floors—or areas. The dataset is organized by individual rooms but can also be processed by considering entire areas at once.

KITTI-360 [33]. This outdoor dataset contains over 100k laser scans acquired in various urban settings on a mobile platform. We use the *accumulated point clouds* format, which consists of large scenes with around 3 million points. There are 239 training scenes and 61 for validation.

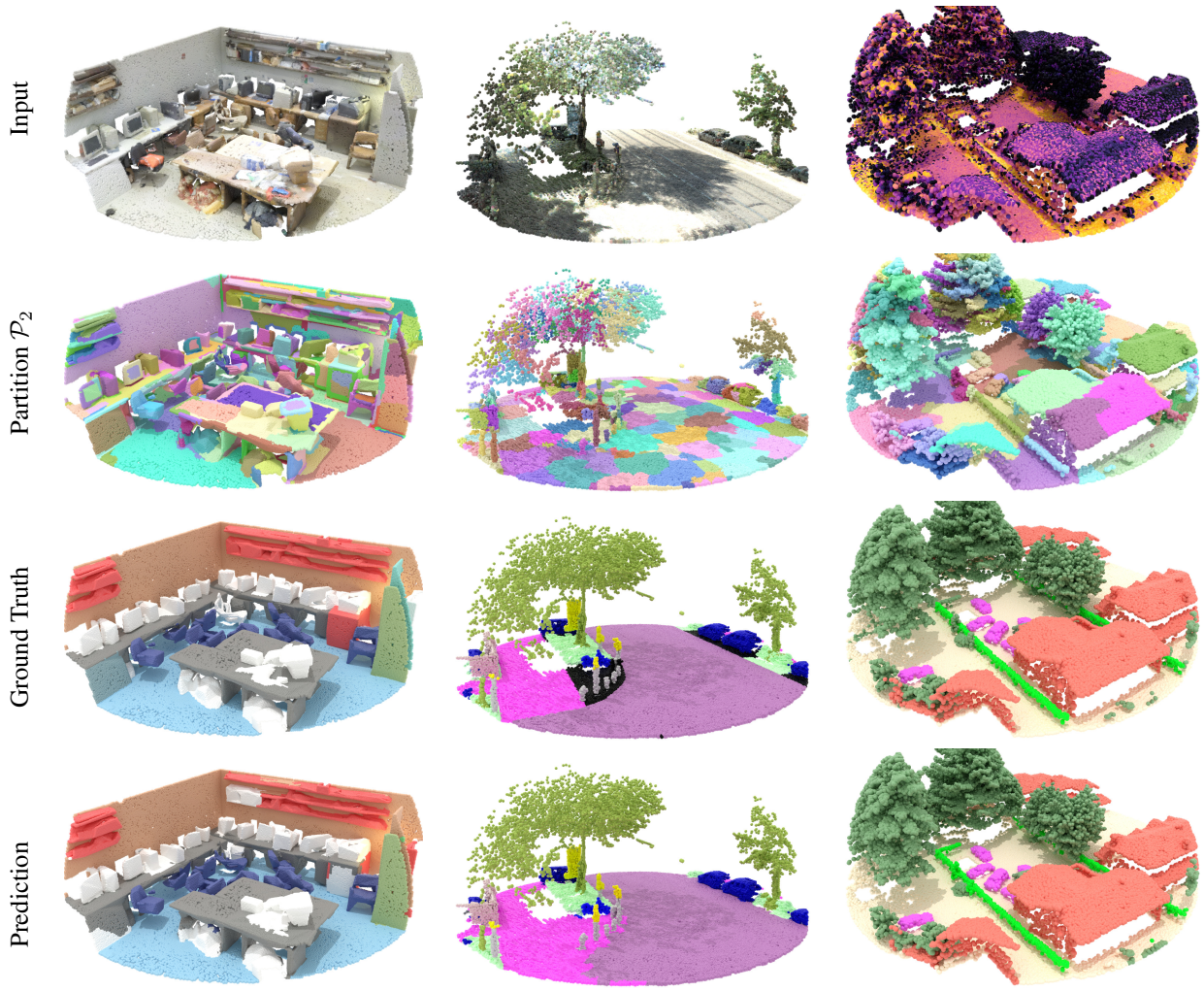
DALES [56]. This 10 km^2 aerial LiDAR dataset contains 500 millions of points across 40 urban and rural scenes, including 12 for evaluation.

We sub-sample the datasets using a 3cm grid for S3DIS, and 10cm for KITTI-360 and DALES. All accuracy metrics are reported for the full, unsampled point clouds. We use a two-level partition ($I = 2$) with $\mu^2 = 50$ for all datasets and select the partition parameters to obtain a 30-fold reduction between $\mathcal{P}_1$ and $\mathcal{P}_0$ and a further 5-fold reduction for $\mathcal{P}_2$. See Table 1 for more details.

Models. We use the same model configuration for all three datasets with minimal adaptations. All transformer modules have a shared width D_{val}, a small key space of dimension $D_{\text{key}} = 4$, 16 heads, with 3 blocks in the encoder and 1 in the decoder. We set $D_{\text{val}} = 64$ for S3DIS and DALES (210k parameters), and $D_{\text{val}} = 128$ (777k parameters) for KITTI360. See the Appendix and our open repository for the detailed configuration of all modules.

We also propose SPT-nano, a lightweight version of our model which does not compute point-level features but operates directly on the first partition level $\mathcal{P}_1$. The value of the maxpool over points in Eq. (1) for $i = 1$ is replaced by f^1, the aggregated handcrafted point features at level 1 of the partition. This model never considers the full point cloud $\mathcal{P}_0$ but only operates on the partitions. For this model, we set $D_{\text{val}} = 16$ for S3DIS and DALES (26k parameters), and $D_{\text{val}} = 32$ for KITTI360 (70k parameters).

Batch Construction. Batches are sampled from large *tiles*: entire building floors for S3DIS, and full scenes for KITTI-



(a) S3DIS (b) KITTI-360 (c) DALES

Figure 4: **Qualitative Results.** We represent input samples (with color or intensity) of our approach and its predictions for all three datasets. Additionally, we show the coarsest partition level and demonstrate how superpoints can accurately capture the contours of complex objects and classify them accordingly. Black points are unlabeled in the ground truth.

360 or DALES. Each batch is composed of 4 randomly-sampled portions of the partition with a radius of 7m for S3DIS and 50m for KITTI and DALES, allowing us to model long-range interactions. During training, we apply a superpoint dropout rate of 0.2 for each superpoint at all hierarchy levels, as well as random rotation, tilting, point jitter and handcrafted features dropout. When sampling points within each superpoint, we set $n_{\text{min}} = 32$ and $n_{\text{max}} = 128$.

Optimization. We use the ADAMW optimizer [38] with default parameters, a weight decay of 10^{-4}, a learning rate of 10^{-2} for DALES and KITTI-360 on and 10^{-1} for S3DIS. The learning rate for attention modules is 10 times smaller than for other weights. Learning rates are warmed up from 10^{-6} for 20 epochs and progressively reduced to 10^{-6} with cosine annealing [39].

4.2. Quantitative Evaluation

Performance Evaluation. As seen in Table 2, SPT performs at the state-of-the-art on two of three datasets despite being a significantly smaller model. On S3DIS, SPT beats PointNeXt-XL with 196× fewer parameters. On KITTI-360, SPT outperforms MinkowskiNet despite a size ratio of 49, and surpasses the performance of the even larger multimodal point-image model DeepViewAgg. On DALES, SPT outperforms ConvPoint by more than 12 points with over 21 times fewer parameters. Although SPT is 1.5 points behind KPConv on this dataset, it achieves these results with 67

Table 2: **Performance Evaluation.** We report the Mean Intersection-over-Union of different methods on three different datasets. SPT performs on par or better than recent methods with significantly fewer parameters. † superpoint-based. ⋆/∗ model with 777k/70k parameters.

Model	Size $\times 10^6$	S3DIS 6-Fold	S3DIS Area 5	KITTI 360 val	DALES
PointNet++ [44]	3.0	56.7	-	-	68.3
† SPG [30]	0.28	62.1	58.0	-	60.6
ConvPoint [4]	4.7	68.2	-	-	67.4
† SPG + SSP [27]	0.29	68.4	61.7	-	-
† SPNet [23]	0.32	68.7	-	-	-
MinkowskiNet [8, 6]	37.9	69.1	65.4	58.3	-
RandLANet [22]	1.2	70.0	-	-	-
KPConv [53]	14.1	70.6	67.1	-	**81.1**
Point Trans.[62]	7.8	73.5	70.4	-	-
RepSurf-U [48]	0.97	74.3	68.9	-	-
DeepViewAgg [50]	41.2	74.7	67.2	62.1	-
Strat. Trans. [26, 59]	8.0	74.9	**72.0**	-	-
PointNeXt-XL [45]	41.6	74.9	71.1	-	-
† **SPT** (ours)	0.21	**76.0**	68.9	**63.5**⋆	79.6
† **SPT-nano** (ours)	**0.026**	70.8	64.9	57.2∗	75.2

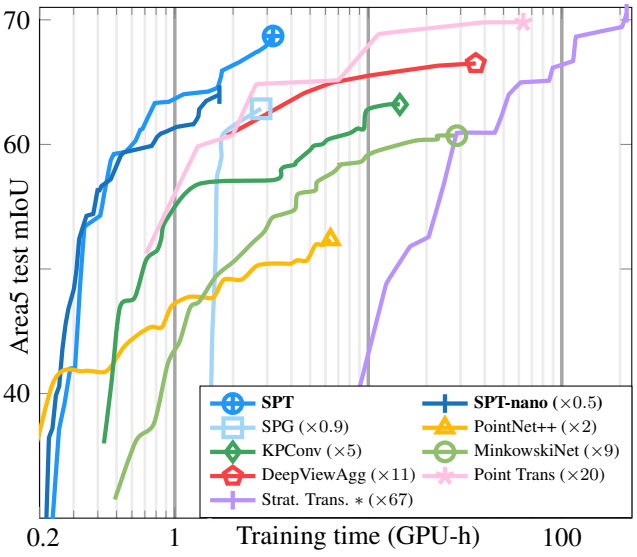


Figure 5: **Training Speed.** We report the evolution of the test mIoU for S3DIS Area 5 for different methods *until the best epoch is reached*. The curves are shifted right according to the preprocessing time. We report in parenthesis the time ratio compared to SPT.

times fewer parameters. SPT achieves significant performance improvements over all superpoint-based methods on all datasets, ranging from 7 to 14 points. SPT overtakes the SSP and SPNet superpoint methods which *learn* the partition

Table 3: **Efficiency Analysis.** We report the preprocessing time for the entire S3DIS dataset and the training and inference time for Area 5. SPT and SPT-nano shows significant speedups in pre-processing, training, and inference times.

	Preprocessing in min	Training in GPU-h	Inference in s
PointNet++ [44]	8.0	6.3	42
KPConv [53]	23.1	14.1	162
MinkowskiNet [8]	20.7	28.8	83
Stratified Trans. [26]	8.0	216.4	30
Superpoint Graph [30]	89.9	1.3	16
SPT (ours)	12.4	3.0	2
SPT-nano (ours)	12.4	1.9	1

in a two-stage training setup leading to preprocessing times of several hours. Interestingly, the lightweight SPT-nano model matches KPConv and MinkowskiNet with only 26k parameters. See figure 4 for qualitative illustrations.

Preprocessing Speed. As reported in Table 3, our implementation of the preprocessing step is highly efficient. We can compute partitions, superpoint-graphs, and handcrafted features, and perform I/O operations quickly: 12.4min for S3DIS, 117 for KITTI-360, and 148 for DALES using a server with a 48-core CPU. An 8-core workstation can preprocess S3DIS in 26.6min. Our preprocessing time is as fast or faster than point-level methods and 7× faster than Superpoint Graph's, thus alleviating one of the main drawbacks of superpoint-based methods.

Training Speed. We trained several state-of-the-art methods from scratch and report in Figure 5 the evolution of the test performance as a function of the training time. We used the official training logs for the multi-GPU Point Transformer and Stratified Transformer. SPT can train much faster than all methods not based on superpoints while attaining similar performance. Although Superpoint Graph trains even faster, its performance saturates earlier, 6.0 mIoU points below SPT . We also report the inference time of our method in Table 3, which is significantly lower than competing approaches, with a speedup factor ranging from 8 to 80. All speed measurements were conducted on a single-GPU server (48 cores, 512Go RAM, A40 GPU). Nevertheless, our model can be trained on a standard workstation (8 cores, 64Go, 2080Ti) with smaller batches, taking only 1.5 times longer and with comparable results.

SPT performs on par or better than complex models with up to two orders of magnitude more parameters and significantly longer training times. Such efficiency and compactness have many benefits for real-world scenarios where hardware, time, or energy may be limited.

Table 4: **Ablation Study.** Impact of some of our design choices on the mIoU for all tested datasets.

Experiment	S3DIS 6-Fold	KITTI 360 Val	DALES
Best Model	76.0	63.5	79.6
a) No handcrafted features	-0.7	-4.1	-1.4
b) No adjacency encoding	-6.3	-5.4	-3.0
b) Fewer edges	-3.5	-1.1	-0.3
c) No point sampling	-1.3	-0.9	-0.5
c) No superpoint sampling	-2.7	-2.5	-0.7
c) Only 1 partition level	-8.4	-5.1	-0.9

4.3. Ablation Study

We evaluate the impact of several design choices in Table 4 and reports here our observations.

a) Handcrafted features. Without handcrafted point features, our model perform worse worse on all datasets. This observation is in line with other works who also remarked the positive impact of well-designed handcrafted features on the performance of smaller models [21, 48].

b) Influence of Edges. Removing the relative positional encoding between superpoints leads to a significant drop of 6.3 points on S3DIS; characterizing the relative position and relationship between superpoints appears crucial for leveraging their context. We also find that pruning the 50% longest edges of each superpoint results in a systematic performance drop, highlighting the importance of modeling long relationships.

c) Partition-Based Improvements. We assess the impact of several improvements made possible by using hierarchical superpoints. First, we find that using all available points when embedding the superpoints of $\mathcal{P}_1$ instead of random sampling resulted in a small performance drop. Second, setting the superpoint dropout rate to 0 worsens the performance by over 2.5 points on S3DIS and KITTI-360.

While we did not observe better results with three or more partition levels, only using one level leads to a significant loss of performance for all datasets.

d) Influence of Partition Purity. In Figure 6, we plot the performance of the "oracle" model which associates to each superpoint of $\mathcal{P}_1$ with its most frequent true label. This model acts as an upper bound on the achievable performance with a given partition. Our proposed partition has significantly higher semantic purity than a regular voxel grid with as many nonempty voxels as superpoints. This implies that our superpoints better adhere to semantic boundaries between objects.

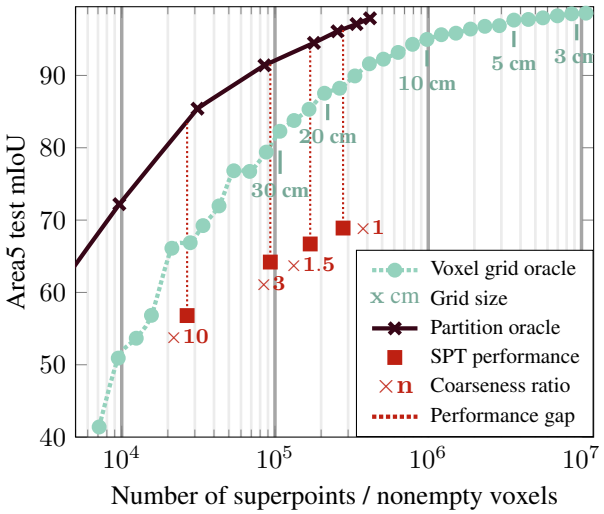


Figure 6: **Partition Purity.** We plot the highest achievable "oracle" prediction for our partitions and a regular voxel grid. We also show the performance of SPT for 4 partitions with a coarseness ratio from $\times 1$ to $\times 10$.

We also report the performance of our model for different partitions of varying coarseness, measured as the number of superpoints in $\mathcal{P}_1$. Using respectively $\times 1.5$ ($\times 3$) fewer superpoints leads to a performance drop of 2.2 (4.7) mIoU points, but reduce the training time to 2.4 (1.6) hours. The performance of SPT is more than 20 points below the oracle, which suggests that the partition does not strongly limit its performance.

Limitations. See Appendix.

5. Conclusion

We have introduced the Superpoint Transformer approach for semantic segmentation of large point clouds, combining superpoints and transformers to achieve state-of-the-art results with significantly reduced training time, inference time, and model size. This approach particularly benefits large-scale applications and computing with limited resources. More broadly, we argue that small, tailored models can offer a more flexible and sustainable alternative to large, generic models for 3D learning. With training times of a few hours on a single GPU, our approach allows practitioners to easily customize the models to their specific needs, enhancing the overall usability and accessibility of 3D learning.

Acknowledgements. This work was funded by ENGIE Lab CRIGEN. This work was supported by ANR project READY3D ANR-19-CE23-0007, and was granted access to the HPC resources of IDRIS under the allocation AD011013388R1 made by GENCI. We thank Bruno Vallet, Romain Loiseau and Ewelina Rupnik for inspiring discussions and valuable feedback.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *TPAMI*, 2012.

[2] Pablo Arbelaez. Boundary extraction in natural images using ultrametric contour maps. *CVPR Workshop*, 2006.

[3] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D semantic parsing of large-scale indoor spaces. *CVPR*, 2016.

[4] Alexandre Boulch. ConvPoint: Continuous convolutions for point cloud processing. *Computers & Graphics*, 2020.

[5] Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-yan Liu, and Liwei Wang. GraphNorm: A principled approach to accelerating graph neural network training. *ICML*, 2021.

[6] Thomas Chaton, Nicolas Chaulet, Sofiane Horache, and Loic Landrieu. Torch-Points3D: A modular multi-task framework for reproducible deep learning on 3D point clouds. *3DV*, 2020.

[7] Shaoyu Chen, Jiemin Fang, Qian Zhang, Wenyu Liu, and Xinggang Wang. Hierarchical aggregation for 3D instance segmentation. *CVPR*, 2021.

[8] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. *CVPR*, 2019.

[9] Jérôme Demantké, Clément Mallet, Nicolas David, and Bruno Vallet. Dimensionality based scale selection in 3D LiDAR point clouds. In *Laserscanning*, 2011.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2020.

[11] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 3d-mpa: Multi-proposal aggregation for 3d semantic instance segmentation. *CVPR*, 2020.

[12] Yuxue Fan, Yan Huang, and Jingliang Peng. Point cloud compression based on hierarchical point clustering. In *APSIPA ASC*, 2013.

[13] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981.

[14] Hongyang Gao and Shuiwang Ji. Graph U-Nets. *ICML*, 2019.

[15] Charlie Giattino, Edouard Mathieu, Julia Broden, and Max Roser. Artificial intelligence. *Our World in Data*, 2022. https://ourworldindata.org/artificial-intelligence.

[16] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. *CVPR*, 2018.

[17] Stéphane Guinard and Loic Landrieu. Weakly supervised segmentation-aided classification of urban scenes from 3d lidar point clouds. *ISPRS Workshop*, 2017.

[18] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. PCT: point cloud transformer. *CVM*, 2021.

[19] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. Occuseg: Occupancy-aware 3D instance segmentation. *CVPR*, 2020.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016.

[21] Pai-Hui Hsu and Zong-Yi Zhuang. Incorporating handcrafted features into deep learning for point cloud classification. *Remote Sensing*, 2020.

[22] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. RandLA-Net: Efficient semantic segmentation of large-scale point clouds. *CVPR*, 2020.

[23] Le Hui, Jia Yuan, Mingmei Cheng, Jin Xie, Xiaoya Zhang, and Jian Yang. Superpoint network for point cloud oversegmentation. *ICCV*, 2021.

[24] James M Joyce. Kullback-Leibler divergence. In *International encyclopedia of statistical science*. Springer, 2011.

[25] Xin Kang, Chaoqun Wang, and Xuejin Chen. Region-enhanced feature learning for scene semantic segmentation. *arXiv preprint arXiv:2304.07486*, 2023.

[26] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3D point cloud segmentation. *CVPR*, 2022.

[27] Loic Landrieu and Mohamed Boussaha. Point cloud oversegmentation with graph-structured deep metric learning. *CVPR*, 2019.

[28] Loic Landrieu and Guillaume Obozinski. Cut pursuit: fast algorithms to learn piecewise constant functions. *AISTATS*, 2016.

[29] Loic Landrieu and Guillaume Obozinski. Cut pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs. In *SIAM Journal on Imaging Sciences*, 2017.

[30] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. *CVPR*, 2018.

[31] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on χ-transformed points. *NeurIPS*, 2018.

[32] Zhihao Liang, Zhihao Li, Songcen Xu, Mingkui Tan, and Kui Jia. Instance segmentation in 3D scenes using semantic superpoint tree networks. *CVPR*, 2021.

[33] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: a novel dataset and benchmarks for urban scene understanding in 2D and 3D. *TPAMI*, 2022.

[34] Yangbin Lin, Cheng Wang, Dawei Zhai, Wei Li, and Jonathan Li. Toward better boundary preserved supervoxel segmentation for 3D point clouds. *ISPRS journal of photogrammetry and remote sensing*, 2018.

[35] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *CVPR*, 2021.

[36] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel CNN for efficient 3D deep learning. *NeurIPS*, 2019.

[37] Romain Loiseau, Mathieu Aubry, and Loïc Landrieu. Online segmentation of lidar sequences: Dataset and algorithm. *ECCV*, 2022.

[38] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[39] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *ICLR*, 2017.

[40] J Narasimhamurthy, Karthikeyan Vaiapury, Ramanathan Muthuganapathy, and Balamuralidhar Purushothaman. Hierarchical-based semantic segmentation of 3D point cloud using deep learning. *Smart Computer Vision*, 2023.

[41] Jeremie Papon, Alexey Abramov, Markus Schoeler, and Florentin Worgotter. Voxel cloud connectivity segmentation-supervoxels for point clouds. *CVPR*, 2013.

[42] Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park. Fast point transformer. *CVPR*, 2022.

[43] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. *CVPR*, 2017.

[44] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NeurIPS*, 2017.

[45] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *NeurIPS*, 2022.

[46] Xingwen Quana, Binbin Hea, Marta Yebrab, Changmin Yina, Zhanmang Liaoa, Xueting Zhanga, and Xing Lia. Hierarchical semantic segmentation of urban scene point clouds via group proposal and graph attention network. *International Journal of Applied Earth Observations and Geoinformation*, 2016.

[47] Hugo Raguet and Loic Landrieu. Parallel cut pursuit for minimization of the graph total variation. *ICML Workshop on Graph Reasoning*, 2019.

[48] Haoxi Ran, Jun Liu, and Chengjie Wang. Surface representation for point clouds. *CVPR*, 2022.

[49] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. *CVPR*, 2017.

[50] Damien Robert, Bruno Vallet, and Loic Landrieu. Learning multi-view aggregation in the wild for large-scale 3D semantic segmentation. *CVPR*, 2022.

[51] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. *MICCAI*, 2015.

[52] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *CVPR*, 2017.

[53] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPConv: Flexible and deformable convolution for point clouds. *ICCV*, 2019.

[54] Anirud Thyagharajan, Benjamin Ummenhofer, Prashant Laddha, Om Ji Omer, and Sreenivas Subramoney. Segment-fusion: Hierarchical context fusion for robust 3D semantic segmentation. *CVPR*, 2022.

[55] Wei-Chih Tu, Ming-Yu Liu, Varun Jampani, Deqing Sun, Shao-Yi Chien, Ming-Hsuan Yang, and Jan Kautz. Learning superpixels with segmentation-aware affinity loss. *CVPR*, 2018.

[56] Nina Varney, Vijayan K Asari, and Quinn Graehling. DALES: A large-scale aerial LiDAR data set for semantic segmentation. *CVPR Workshops*, 2020.

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

[58] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *ICLR*, 2018.

[59] Qi Wang, Shengge Shi, Jiahui Li, Wuming Jiang, and Xiangde Zhang. Window normalization: Enhancing point cloud understanding by unifying inconsistent point densities. 2022.

[60] Yongchao Xu, Thierry Géraud, and Laurent Najman. Hierarchical image simplification and segmentation based on mumford–shah-salient level line selection. *Pattern Recognition Letters*, 2016.

[61] Zizhao Zhang, Han Zhang, Long Zhao, Ting Chen, Sercan Ö Arik, and Tomas Pfister. Nested hierarchical transformer: Towards accurate, data-efficient and interpretable visual understanding. *AAAI*, 2022.

[62] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. *ICCV*, 2021.

Appendix

In this document, we introduce our interactive visualization tool (Section A-1), share our source code (Section A-2), discuss limitations of our approach (Section A-3), provide a description (Section A-4) and an analysis (Section A-5) of all handcrafted features used by our method, detail the construction of the superpoint-graphs (Section A-6) and the partition process (Section A-7), and provide guidelines on how to choose the partition's hyper-parameters (Section A-8). Finally, we clarify our architecture parameters (Section A-9), explore our model's salability (Section A-10) and supervision (Section A-11), detail the class-wise performance of our approach on each dataset (Section A-12), as well as the color maps used in the illustrations of the main paper (Figure A-3).

A-1. Interactive Visualization

We release for this project an interactive visualization tool which produces HTML files compatible with any browser. As shown in Figure A-1, we can visualize samples from S3DIS, KITTI-360, and DALES with different point attributes and from any angle. These visualizations were instrumental in designing and validating our model, and we hope that they will be help the reader's understanding as well.

A-2. Source Code

We make our source code publicly available at `github.com/drprojects/superpoint_transformer`. The code provides all necessary instructions for installing and navigating the project, simple commands to reproduce our main results on all datasets, off-the-shelf pretrained models, and ready-to-use notebooks. Our method is developed in Pytorch and relies on the libraries PyTorch Geometric, PyTorch Lightning, and Hydra.

A-3. Limitations

Our model provides significant advantages in terms of speed and compacity but also comes with its own set of limitations.

Overfitting and Scaling. The superpoint approach drastically simplifies and compresses the training sets: the 274m 3D points of S3DIS are captured by a geometry-driven multi-level graph structure with fewer than 1.25m nodes. While this simplification favours model compacity and training speed, this can lead to overfitting when using SPT configurations with more parameters, as shown in Section A-10. Scaling our model to millions of parameters may only yield better results for training sets that are sufficiently large, diverse, and complex.

Errors in the Partition. Object boundaries lacking obvious discontinuities, such as curbs vs. roads or whiteboards vs. walls, are not well recovered by our partition. As partition errors cannot be corrected by our approach, this may lead to classification errors. To improve this, we could replace our handcrafted point descriptors (Section A-4) with features directly learned for partitioning [27, 23]. However, such methods significantly increase the preprocessing time, contradicting our current focus on efficiency. In line with [21, 48], we use easy-to-compute yet expressive handcrafted features. Our model SPT-nano without point encoder relies purely on such features and reaches 70.8 mIoU on S3DIS 6-Fold with only 27k param, illustrating this expressivity.

Learning Through the Partition. The idea of learning point and adjacency features end-to-end directly is a promising research direction to improve our model. However, this implies efficiently backpropagating through superpoint hard assignments, which remains an open problem. Furthermore, such a method would consider individual 3D points during training, which may negate the efficiency of our method.

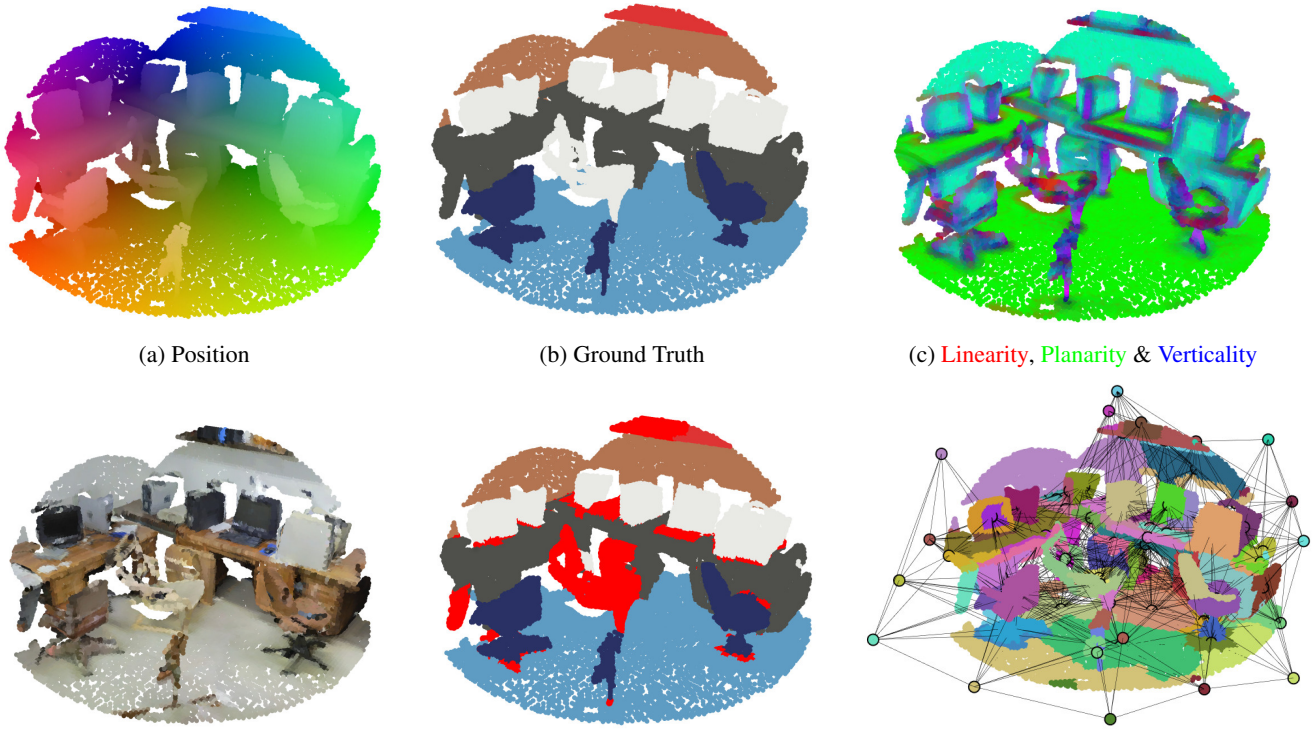
Point-Level Prediciton. Finally, our method predicts labels at the superpoint level P_1 and not individual 3D points. Since this may limit the maximum performance achievable by our approach, we could consider adding an upsampling layer to make point-level predictions. However, this does not appear to us as the most profitable research direction. Indeed, this may negate some of the efficiency of our method. Furthermore, as shown in the ablation study 4.3 d) of the main paper, the "oracle" model outperforms ours by a large margin. This may indicate that performance improvements should primarily be searched in superpoint classification rather than improving the partition.

Our model also learns features for superpoints and not individual 3D points. This may limit downstream tasks requiring 3D point features such as surface reconstruction, or panoptic segmentation. However, we argue that specific adaptations could be explored to perform these tasks at the superpoint level.

A-4. Handcrafted Features

Our method relies on simple handcrafted features to build the hierarchical partition and learn meaningful points and adjacency relationships. In this section, we provide further details on the definition of these features and how to compute them. It is important to note that these features are only computed once during preprocessing, and thanks to our optimized implementation, this step only takes a few minutes.

Point Features. We can associate each 3D point with a set 8 of easy-to-compute handcrafted features, described below.



(a) Position (b) Ground Truth (c) Linearity, Planarity & Verticality

(d) RGB (e) Predictions & Errors (f) Level-2

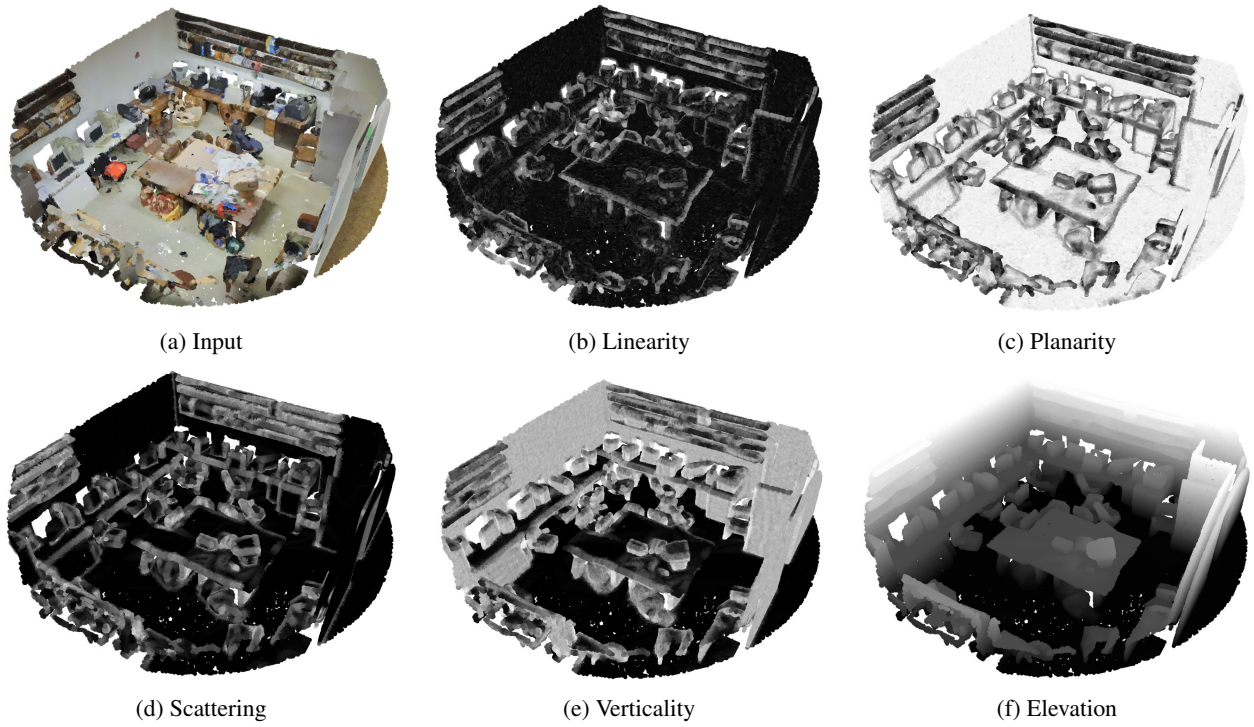
Figure A-1: **Interactive Visualization.** Our interactive viewing tool allows for the manipulation and visualization of sample point clouds colorized according to their position (a), semantic labels (b), selected geometric features (c), radiometry (d), and to visualize our network's prediction (e) and partitions (f).

- *Radiometric features* (3 or 1): RGB colors are available for S3DIS and KITTI-360, and intensity values for DALES. These radiometric features are normalized to $[0, 1]$ at preprocessing time. For KITTI-360, we find that using the HSV color model yields better results.
- *Geometric features* (5): We use PCA-based features: *linearity*, *planarity*, *scattering*, [9] and *verticality* [17], computed on the set of 50-nearest neighbors of each point. This neighbor search is only computed once during preprocessing and is also necessary to build the graph $\mathcal{G}$. We also define the *Elevation* as the distance between a point and the ground below it. Since the ground is not necessarily flat nor horizontal, we use the RANSAC algorithm [13] on a coarse subsampling of the scene to find a ground plane. We normalize the elevation by dividing it by 4 for S3DIS and 20 for DALES and KITTI-360.

At preprocessing time, we only use radiometric and geometric features to compute the hierarchical partition. At training time, SPT computes point embeddings by mapping all available point features, along with the normalized point position to a vector of size D_{point} with a dedicated MLP ϕ^0_{enc}. We provide an illustration of the point geometric features in Figure A-2, to help the reader apprehend these simple geometric descriptors.

Adjacency Features. The relationship between adjacent superpoints provides crucial information to leverage their context. For each edge of the superpoint-graph, we compute the 18 following features:

- *Interface features (7)*: All adjacent superpoints share an *interface*, *i.e.* pairs of points from each superpoint that are close and share a line of sight. SuperpointGraph [30] uses the Delaunay triangulation of the entire point cloud to compute such interfaces, while we propose a faster heuristic approach in Section A-6. Each pair of points of an interface defines an offset, *i.e.* a vector pointing from one superpoint to its neighbor. We compute the mean offset (dim 3), the mean offset length (dim 1), and the standard deviation of the offset in each canonical direction (dim 3).
- *Ratio features (4)*: As defined in [30], we characterize each pair of adjacent superpoints with the ratio of their *lengths*, *surfaces*, *volumes*, and *point counts*.



(a) Input (b) Linearity (c) Planarity

(d) Scattering (e) Verticality (f) Elevation

Figure A-2: **Point Geometric Features.** Given an input cloud (a), the computed PCA-based geometric features (b, c, d, e) and distance to the ground (f) offer a simple characterization of the local geometry around each point.

Table A-1: **Ablation on Handcrafted Features.** Impact of handcrafted features on the mIoU for all tested datasets.

Experiment	S3DIS 6-Fold	KITTI 360 Val	DALES
Best Model	76.0	63.5	79.6
a) Point Features			
No radiometric feat.	-2.7	-4.0	-1.2
No geometric feat.	-0.7	-4.1	-1.4
b) Adjacency Features			
No interface feat.	-0.2	-0.6	-0.7
No ratio feat.	-1.1	-2.2	-0.4
No pose feat.	-5.5	-1.2	-0.8
c) Room Features			
Room-level samples	-3.8	-	-
Room position	-0.7	-	-

- *Pose features (7)*: We characterize the relative position between two superpoints with the cosine of the angle between the superpoint normal vectors (dim: 1) and between each of the two superpoints' normal and the mean offset direction (dim: 2). Additionally, the offset between the centroids of the superpoints is used to compute the centroid distance (dim: 1) and the unit-normalized centroid offset direction (dim: 3).

Note that the mean offset and the ratio features are not symmetric and imply that the edges of the superpoint-graphs are oriented. As mentioned in Section 3.3, a network ϕ^i_{adj} maps these handcrafted features to a vector of size D_{adj}, for each level $i \geq 1$.

A-5. Influence of Handcrafted Features

In Table A-1, we quantify the impact of the handcrafted features detailed in Section A-4 on performance. To this end, we retrain SPT without each feature group and evaluate the prediction on S3DIS Area 5.

a) Point Features. Our experiments show that removing radiometric features has a strong impact on performance, with a drop of 2.7 to 4.0 mIoU. In contrast, removing geometric features results in a performance drop of 0.7 on S3DIS, but 4.1 on KITTI-360.

We observe that both outdoor datasets strongly benefit from local geometric features, which we hypothesize is due to their lower resolution and noise level. These results indicate that radiometric features play an important role for all datasets and that geometric features may facilitate learning on noisy or subsampled datasets.

b) Adjacency Features. The analysis of the impact of adjacency features on our model's performance indicates that they play a crucial role in leveraging contextual information from superpoints: removing all adjacency features leads to a significant drop of 3.0 to 6.3 mIoU points on the datasets, as shown in 4.3 b) of the main paper. Among the different types of adjacency features, pose features appear particularly useful in characterizing the adjacency relationships between superpoints of S3DIS, while interface features have a smaller impact. These results suggest that the relative pose of objects in the scene may have more influence on the 3D semantic analysis performed by our model than the precise characterization of their interface. On the other hand, interface and ratio features seem to have more impact on outdoor datasets, while the pose information seems to be less informative in the semantic understanding of the scene.

c) S3DIS Room Partition. The S3DIS dataset is divided into individual rooms aligned along the x and y axes. This setup simplifies the classification of classes such as walls, doors, or windows as they are consistently located at the edge of the room samples. Some methods also add normalized room coordinates to each points. However, we argue that this partition may not generalize well to other environments, such as open offices, industrial facilities, or mobile mapping acquisitions, which cannot naturally be split into rooms.

To address this limitation, we use the absolute room positions to reconstruct the entire floor of each S3DIS area [53, 6]. This enables our model to consider large multi-room samples, resulting in a performance increase of 3.8 points. This highlights the advantage of capturing long-range contextual information. Additionally, we remark that SPT performs better without using room-normalized coordinates, which may lead to overfitting and poor performance on layouts that deviate from the room-based structure of the S3DIS dataset such as large amphitheaters.

A-6. Superpoint-Graphs Computation

The Superpoint Graph method by Landrieu and Simonovsky [30] builds a graph from a point cloud using Delaunay triangulation, which can take a long time for large point clouds. In contrast, our approach connects two superpoints in $\mathcal{P}_i$, where $i \geq 1$ if their closest points are within a distance gap $\epsilon_i > 0$. However, computing pairwise distances for all points is computationally expensive. We propose a heuristic to approximately find the closest pair of points for two superpoints, see Algorithm A-1. We also accelerate the computation of adjacent superpoints by approximating only for superpoints with centroids closer than the sum of their radii plus the gap distance. This approximation helps to reduce the number of computations required for adjacency computation, which leads to faster processing times. All steps involved in the computation of our superpoint-graph are implemented on the GPU to further enhance computational efficiency.

Algorithm A-1 Approximate Superpoint Gap

Input: superpoints p_1 and p_2, num_steps
$c_1 \leftarrow \text{centroid}(p_1)$
$c_2 \leftarrow \text{centroid}(p_2)$
for $s \in$ num_steps **do**
  $c_2 \leftarrow \arg\min_{p \in p2} \|c_1 - p\|$
  $c_1 \leftarrow \arg\min_{p \in p1} \|c_2 - p\|$
end for
  return $\|c_1 - c_2\|$

Recovering the interface between two adjacent superpoints as evoked in Section A-4 involves a notion of visibility: we connect points from each superpoint which are *facing* each other. This can be a challenging and ambiguous problem, which SuperPoint Graph [28] tackles using a Delaunay triangulation of the points. However, this method is impractical for large point clouds. To address this issue, we propose a heuristic approach with the following steps: (i) first, we use the Approximate Superpoint Gap algorithm to compute the approximate nearest points for each superpoint. Then, we restrict the search to only consider points within a certain distance of the nearest points. Finally, we match the points on each side of the interface based on their order along a vector perpendicular to the direction connecting the two nearest points.

A-7. Details on Hierarchical Partitions

We present here a more detailed explanation of the hierarchical partition process. We define for each point c of $\mathcal{C}$ a feature f_c of dimension D, and $G := (\mathcal{C}, \mathcal{E}, w)$ is the k-nn adjacency between the points, with $w \in \mathbb{R}_+^{\mathcal{E}}$ a nonnegative proximity value. Our goal is to compute a multi-level hierarchical partition of the point cloud in superpoints homogeneous with respect to f at increasing coarseness.

Piecewise Constant Approximation on a Graph. We first explain how to compute a single-level partition of the point cloud. We consider the pointwise features f_c as a D-dimensional signal $f \in \mathbb{R}^{D \times \mathcal{C}}$ defined on the nodes of the weighted graph $G := (\mathcal{C}, \mathcal{E}, w)$. We first define an energy $\mathcal{J}(e; f, \mathcal{G}, \lambda)$ measuring the fidelity between a vertex-valued signal $e \in \mathbb{R}^{D \times \mathcal{C}}$ and the length of its contours, defined as the weight of the cut between its constant components [28]:

$$\mathcal{J}(e; f, \mathcal{G}, \lambda) := \|e - f\|^2 + \lambda \sum_{(u,v) \in \mathcal{E}} w_{u,v} \left[e_u \neq e_v\right] \ , \tag{A-6}$$

with $\lambda \in \mathbb{R}_+$ a regularization strength and $[a \neq b]$ the function equals to 0 if $a = b$ and 1 otherwise. Minimizers of

Table A-2: **Model Configuration.** We provide the detailed architecture of the SPT-X architecture. In this paper, we use $X = 64$ and $X = 128$.

Parameter	Value
Handcrafted features	
$D^{\text{hf}}_{\text{point}}$	$D^{\text{radio}}_{\text{point}} + D^{\text{geof}}_{\text{point}}$
$D^{\text{hf}}_{\text{adj}}$	18
Embeddings sizes	
D_{point}	128
D_{adj}	32
Transformer blocks	
D_{val}	**X**
D_{key}	4
# blocks encoder	3
# blocks decoder	1
# heads	16
MLPs	
ϕ^{i}_{adj}	$[D^{\text{hf}}_{\text{adj}}, D_{\text{adj}}, D_{\text{adj}}, 3D_{\text{adj}}]$
ϕ^{0}_{enc}	$[D^{\text{hf}}_{\text{point}} + D^{\text{pos}}_{\text{point}}, 32, 64, D_{\text{point}}]$
ϕ^{1}_{enc}	$[D_{\text{point}} + D^{\text{pos}}_{\text{point}}, D_{\text{val}}, D_{\text{val}}]$
ϕ^{2}_{enc}	$[D_{\text{val}} + D^{\text{pos}}_{\text{point}}, D_{\text{val}}, D_{\text{val}}]$
ϕ^{1}_{dec}	$[D_{\text{val}} + D_{\text{val}} + D^{\text{pos}}_{\text{point}}, D_{\text{val}}, D_{\text{val}}]$

$\mathcal{J}$ are approximations of f that are piecewise constant with respect to a partition with simple contours in $\mathcal{G}$.

We can characterize such signal $e \in \mathbb{R}^{D \times \mathcal{C}}$ by the coarsest partition $\mathcal{P}^e$ of $\mathcal{N}$ and its associated variable $f^e \in \mathbb{R}^{D \times \mathcal{P}^e}$ such that e is constant within each segment p of P^e with value f^e_p. The partition $\mathcal{P}^e$ also induces a graph $\mathcal{G}^e := (\mathcal{P}^e, \mathcal{E}^e, w^e)$ with $\mathcal{E}^e$ linking the component of $\mathcal{P}^e$ adjacent in $\mathcal{G}$ and w^e the weight of the cut between adjacent elements of P^e:

$$\mathcal{E}^e := \{(U, V) \mid U, V \in \mathcal{P}^e, (U \times V) \cap \mathcal{E} \neq \varnothing\} \quad \text{(A-7)}$$

$$w^e_{U,V} := \sum_{(u,v) \in U \times V \cap \mathcal{E}} w_{u,v} \text{ for } (U, V) \in \mathcal{E}^e \quad \text{(A-8)}$$

We denote by $\text{partition}\,(e)$ the function mapping e to these uniquely defined variables:

$$f^e, P^e, \mathcal{G}^e := \text{partition}\,(e)\ . \quad \text{(A-9)}$$

Note that, in this section, $\mathcal{G}^e$ is *not* the superpoint-graph used for self-attention in the SPT architecture.

Point Cloud Hierarchical Partition. A set of partitions $\mathcal{P} := [\mathcal{P}_0, \cdots, \mathcal{P}_i]$ defines a hierarchical partition of $\mathcal{C}$ with I levels if $\mathcal{P}_0 = \mathcal{C}$ and P_{i+1} is a partition of P_i for $i \in [0, I-1]$. We propose to use the formulations above to define a hierarchical partition of the point cloud $\mathcal{C}$ characterized by a list $\lambda_1, \cdots, \lambda_I$ of non-decreasing non-negative regularization strengths defining the coarseness of the successive partitions. We first define $\mathcal{G}_0$ as the point-level adjacency graph $\mathcal{G}$ and f_0 as f. We can now define the levels of a hierarchical partition $\mathcal{P}$ for $i \in [1, I]$:

$$f_{i+1}, \mathcal{P}_{i+1}, \mathcal{G}_{i+1} := \text{partition}(\underset{e \in \mathbb{R}^{D \times \mathcal{P}_i}}{\arg\min}\, \mathcal{J}\,(e; f_i, \mathcal{G}_i, \lambda_i)). \quad \text{(A-10)}$$

Given that the optimization problems defined in Eq. (A-10) for $i > 1$ operate on the component graphs $\mathcal{G}_i$, which are smaller than $\mathcal{G}_0$, the first partition is the most demanding in terms of computation.

A-8. Parameterizing the Partition

We define $\mathcal{G}$ as the $k = 10$-nearest neighbor adjacency graph and set all edge weights w to 1. The point features f_p whose piecewise constant approximation yields the partition are of three types: geometric, radiometric, and spatial.

Geometric features ensure that the superpoints are geometrically homogeneous and with simple shapes. We use the normalized dimensionality-based method described in Section A-4. Radiometric features encourage the border of superpoints to follow the color contrast of the scene, and are either RGB or intensity values; they must be normalized to fall in the [0,1] range. Lastly, we can add to each point their spatial coordinate with a normalization factor μ in m^{-1} to limit the size of superpoints. We recommend to set μ as the inverse of the maximum radius expected for a superpoint: the largest sought object (facade, wall, roof) or an application-dependent constraint.

The coarseness of the partition depends on the regularization strength λ as defined in Section A-6. Finer partitions should generally lead to better results, but an increase in training time and memory requirement. We chose a ratio $\mid \mathcal{P}_0 \mid / \mid \mathcal{P}_1 \mid \sim 30$ across all datasets as it proved to be a good compromise between efficiency and precision. Depending on the desired trade-off, different ratios can be chosen by trying other values of λ.

A-9. Implementation Details

We provide the exact parameterization of the SPT architecture used for our experiments. All MLPs in the architecture use LeakyReLU activations and GraphNorm [5] normalization. For simplicity, we represent an MLP by the list of its layer widths: [in_channels, hidden_channels, out_channels].

Point Input Features. We refer here to the dimension of point positions, radiometry, and geometric features as $D^{\text{pos}}_{\text{point}} = 3$, $D^{\text{radio}}_{\text{point}}$, and $D^{\text{geof}}_{\text{point}}$ respectively. As seen in Section A-4, S3DIS and KITTI-360 use $D^{\text{radio}}_{\text{point}} = 3$, while DALES uses $D^{\text{radio}}_{\text{point}} = 1$.

Model Architecture. The exact architecture SPT-64 used for S3DIS and DALES is detailed in Table A-2. The other evaluated models are SPT-16, SPT-32, SPT-128 (used for KITTI-360), and SPT-256, which use the same parameters except for D_{val}.

SPT-nano. For SPT-nano, we use and $D_{\text{val}} = 16$, $D_{\text{adj}} = 16$, and $D_{\text{key}} = 2$. As SPT-nano does not compute point embedding it does use ϕ^0, and we set up ϕ^1_{enc} as $[D^{\text{hf}}_{\text{point}} + D^{\text{pos}}_{\text{point}}, D_{\text{val}}, D_{\text{val}}]$.

A-10. Model Scalability

We study the scalability of SPT by comparing models with different parameter counts on each dataset. It is important to note that the superpoint approach drastically compresses the training set, which can lead to overfitting, see Section A-3. For instance, as illustrated in Table A-3, SPT-128 with $D_{\text{val}} = 128$ (777k param.) performs 1.4 points below $D_{\text{val}} = 64$ on S3DIS.

We report a similar behaviour for other hyperparameters: in Table A-4, $D_{\text{key}} = 8$ instead of 4 incurs a drop of 1.0, while in Table A-5, $N_{\text{heads}} = 32$ instead of 16 a drop of 0.1 point. For the larger KITTI-360 dataset (13m nodes), $D_{\text{val}} = 128$ performs 1.9 points above $D_{\text{val}} = 64$, but 5.4 points above $D_{\text{val}} = 256$ (2.7m param.).

Table A-3: **Ablation on Model Scaling.** Impact of model size for each dataset.

Model	Size $\times 10^6$	S3DIS 6-Fold	KITTI 360 Val	DALES
SPT-32	0.14	74.5	60.6	78.7
SPT-64	0.21	**76.0**	61.6	**79.6**
SPT-128	0.77	74.6	**63.5**	78.8
SPT-256	1.80	74.0	58.1	77.6

Table A-4: **Ablation on Query-Key Dimension.** Impact of D_{key} on S3DIS 6-Fold.

D_{key}	2	4	8	16
SPT-64	75.6	**76.0**	75.0	74.7

Table A-5: **Ablation on Heads Count.** Impact of the number of heads N_{head} on the S3DIS 6-Fold performance.

N_{head}	4	8	16	32
SPT-64	74.3	75.2	**76.0**	75.9

A-11. Hierarchical Supervision

We explore, in Table A-6, alternatives to our hierarchical supervision (cross entropy for $\mathcal{P}_1$ and KL for $\mathcal{P}_2$) introduced in Section 3.3. We use "CE-$\mathcal{P}_i$" to refer to cross-entropy supervision on the dominant label applied to the $\mathcal{P}_i$ partition. Similarly, "KL-$\mathcal{P}_i$" denotes the Kullback–Leibler divergence supervision on the distribution of labels within each superpoint of the partition $\mathcal{P}_i$.

We observe a consistent improvement across all datasets by adding the KL-$\mathcal{P}_2$ supervision. This illustrates the benefits of supervising higher-level partitions, despite their lower purity. Moreover, supervising $\mathcal{P}_1$ with KL rather than CE leads to a further performance drop. This validates our choice of considering $\mathcal{P}_1$ superpoints to be sufficiently pure to be supervised using their dominant label.

Table A-6: **Ablation on Supervision.** Impact of our hierarchical supervision for each dataset.

Loss	S3DIS 6-Fold	KITTI 360 Val	DALES
CE-$\mathcal{P}_1$ KL-$\mathcal{P}_2$	**76.0**	**63.5**	**79.6**
CE-$\mathcal{P}_1$	-0.2	-0.8	-0.8
KL-$\mathcal{P}_1$	-0.8	-1.3	-0.8

A-12. Detailed Results

We report in Table A-7 the class-wise performance across all datasets for SPT and other methods for which this information was available. As previously stated, SPT performs close to state-of-the-art methods on all datasets, while being significantly smaller and faster to train. By design, superpoint-based methods can capture long-range interactions, and their predictions are more spatially regular than point-based approaches. This may explain the performance of SPT on in S3DIS, which encompass large, geometrically homogeneous objects or whose identification requires long-range context understanding, such as ceiling, floor, columns, and windows For all datasets, results show that some progress could be made in analyzing smaller objects with intricate geometries. This suggests that a more powerful point-level encoding may be beneficial.

Table A-7: **Class-wise Performance.** Class-wise mIoU across all datasets for our Superpoint Transformer .

S3DIS Area 5

Method	mIoU	ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter
PointNet [43]	41.1	88.8	97.3	69.8	**0.1**	3.9	46.3	10.8	52.6	58.9	40.3	5.9	26.4	33.2
SPG [30]	58.4	89.4	96.9	78.1	0.0	42.8	48.9	61.6	84.7	75.4	69.8	52.6	2.1	52.2
MinkowskiNet [8]	65.4	91.8	**98.7**	86.2	0.0	34.1	48.9	62.4	81.6	**89.8**	47.2	74.9	74.4	58.6
SPG + SSP [27]	61.7	91.9	96.7	80.8	0.0	28.8	60.3	57.2	85.5	76.4	70.5	49.1	51.6	53.3
KPConv [53]	67.1	92.8	97.3	82.4	0.0	23.9	58.0	69.0	91.0	81.5	75.3	75.4	66.7	58.9
PointTrans.[62]	70.4	94.0	98.5	**86.3**	0.0	38.0	63.4	74.3	89.1	82.4	74.3	80.2	76.0	59.3
DeepViewAgg [50]	67.2	87.2	97.3	84.3	0.0	23.4	**67.6**	72.6	87.8	81.0	76.4	54.9	**82.4**	58.7
Stratified PT [26]	**72.0**	**96.2**	**98.7**	85.6	0.0	**46.1**	60.0	**76.8**	**92.6**	84.5	**77.8**	75.2	78.1	**64.0**
SPT	68.9	92.6	97.7	83.5	**0.2**	42.0	60.6	67.1	88.8	81.0	73.2	**86.0**	63.1	60.0
SPT-nano	64.9	92.4	97.1	81.6	0.0	38.2	56.4	58.6	86.3	77.3	69.6	82.5	50.5	53.4

S3DIS 6-FOLD

PointNet [43]	47.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	42.0	54.1	38.2	9.6	29.4	35.2
SPG [30]	62.1	89.9	95.1	76.4	62.8	47.1	55.3	68.4	73.5	69.2	63.2	45.9	8.7	52.9
ConvPoint [4]	68.2	**95.0**	**97.3**	81.7	47.1	34.6	63.2	73.2	75.3	71.8	64.9	59.2	57.6	65.0
MinkowskiNet [8, 50]	69.5	91.2	90.6	83.0	59.8	52.3	63.2	75.7	63.2	64.0	69.0	72.1	60.1	59.2
SPG + SSP [27]	68.4	91.7	95.5	80.8	62.2	54.9	58.8	68.4	78.4	69.2	64.3	52.0	54.2	59.2
KPConv [53]	70.6	93.6	92.4	83.1	63.9	54.3	66.1	76.6	57.8	64.0	69.3	74.9	61.3	60.3
DeepViewAgg [50]	74.7	90.0	96.1	**85.1**	66.9	56.3	**71.9**	**78.9**	79.7	73.9	**69.4**	61.1	**75.0**	**65.9**
SPT	**76.0**	93.9	96.3	84.3	**71.4**	**61.3**	70.1	78.2	**84.6**	**74.1**	67.8	**77.1**	63.6	65.0
SPT-nano	70.8	93.1	96.0	80.9	68.4	54.0	62.2	71.3	76.3	70.8	63.3	74.3	51.9	57.6

KITTI-360 Val

Method	mIoU	road	sidewalk	building	wall	fence	pole	traffic lig.	traffic sig.	vegetation	terrain	person	car	truck	motorcycle	bicycle
MinkowskiNet [8, 50]	54.2	90.6	74.4	84.5	45.3	42.9	52.7	0.5	38.6	87.6	70.3	26.9	87.3	66.0	28.2	17.2
DeepViewAgg [50]	57.8	**93.5**	77.5	89.3	53.5	**47.1**	**55.6**	18.0	44.5	**91.8**	71.8	40.2	87.8	30.8	39.6	**26.1**
SPT	**63.5**	93.3	**79.3**	**90.8**	**56.2**	45.7	52.8	**20.4**	**51.4**	89.8	**73.6**	**61.6**	**95.1**	**79.0**	**53.1**	10.9
SPT-nano	57.2	91.7	74.7	87.8	49.3	38.8	49.0	12.2	39.2	88.0	69.5	39.9	94.2	80.1	33.7	10.4

DALES

Method	mIoU	ground	vegetation	car	truck	power line	fence	pole	building
PointNet++ [44]	68.3	94.1	91.2	75.4	30.3	79.9	46.2	40.0	89.1
ConvPoint [4]	67.4	96.9	91.9	75.5	21.7	86.7	29.6	40.3	96.3
SPG [30]	60.6	94.7	87.9	62.9	18.7	65.2	33.6	28.5	93.4
PointCNN [31]	58.4	**97.5**	91.7	40.6	40.8	26.7	52.6	57.6	95.7
KPConv [53]	**81.1**	97.1	**94.1**	85.3	41.9	**95.5**	**63.5**	**75.0**	96.6
SPT	79.6	96.7	93.1	**86.1**	**52.4**	94.0	52.7	65.3	**96.7**
SPT-nano	75.2	96.5	92.6	78.1	35.8	92.1	50.8	59.9	96.0

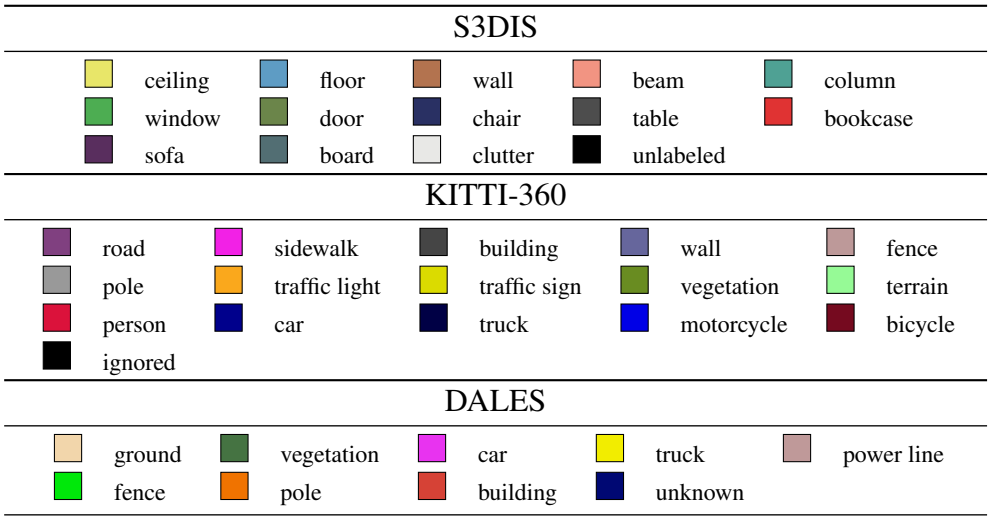


Figure A-3: **Colormaps.**