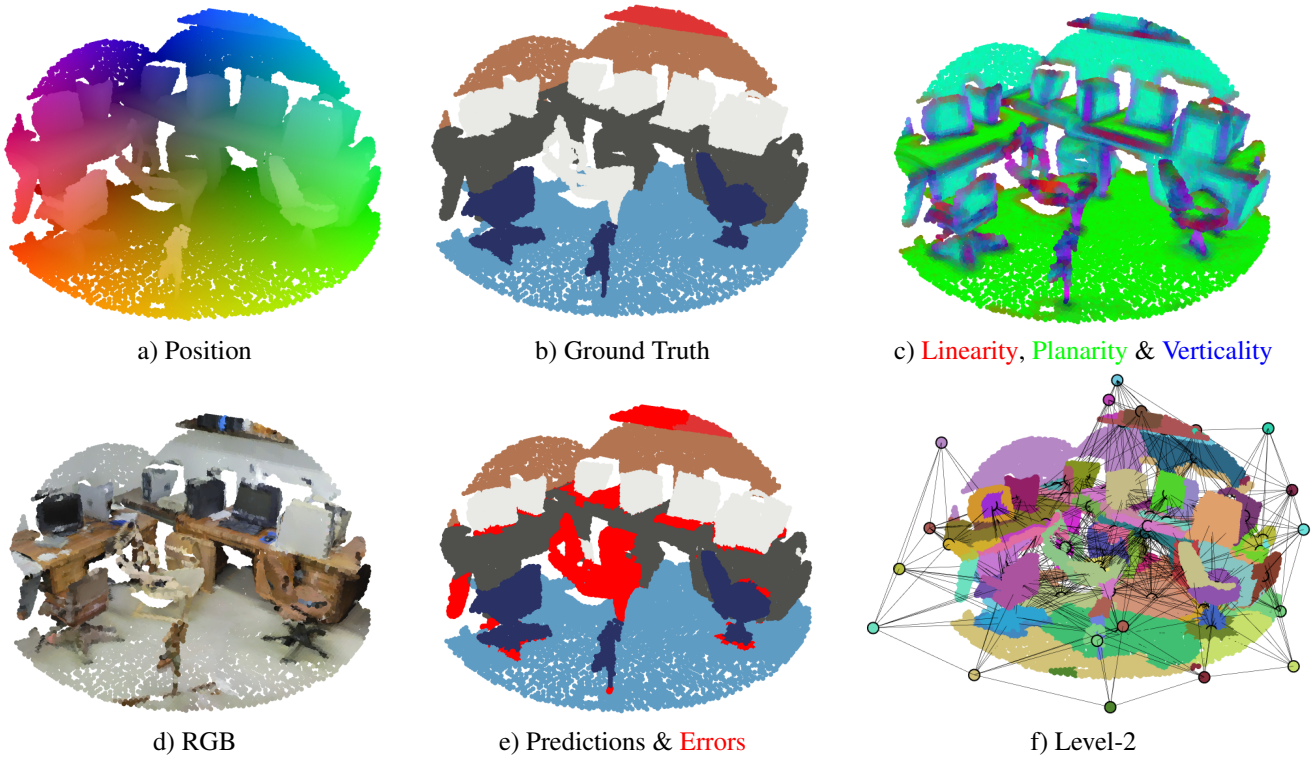
Supplementary Material for: Efficient 3D Semantic Segmentation with Superpoint Transformer

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

[1]CSAI, ENGIE Lab CRIGEN, Stains, France
[2] Univ Gustave Eiffel, IGN/ENSG, LASTIG, F-77454 Marne-la-Vallee, France
[3]INSA Centre Val-de-Loire Univ de Tours, LIFAT, France

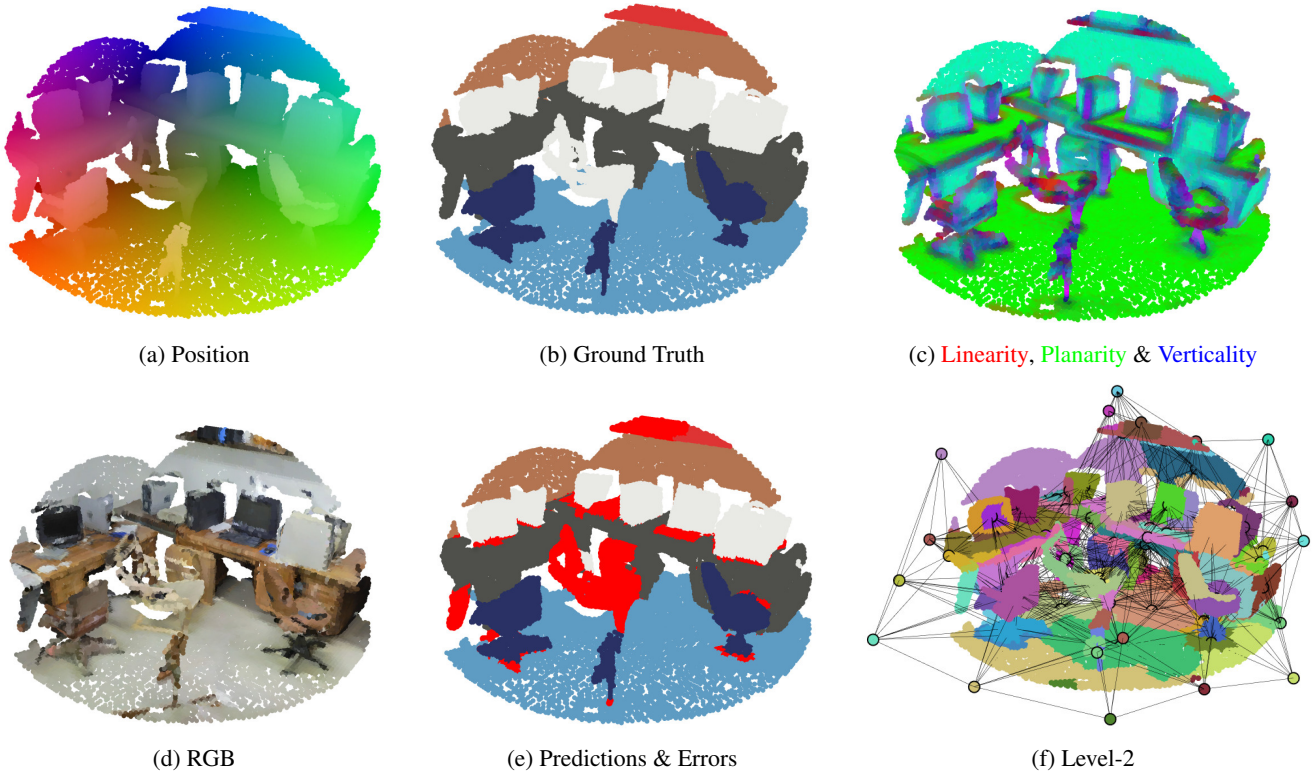


a) Position b) Ground Truth c) Linearity, Planarity & Verticality

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

Figure 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).

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 hyperparameters (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), and the color maps used in the illustrations of the main paper (Figure A-3).



(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).

A-1. Interactive Visualization

We release for this project an interactive plotly visualization tool that 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 facilitate 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, ready-to-use pretrained models, and ready-to-use notebooks.

Our method is developed in PyTorch and relies on 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 multilevel graph structure with fewer than 1.25m nodes. While this simplification favors the compacity and speed of the training of the model, 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 with 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 [11, 9]. However, such methods significantly increase the preprocessing time, contradicting our current focus on efficiency. In line with [8, 17], 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 directly end-to-end 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 would necessitate to perform the partitioning step multiple times during training time, which may negate the efficiency of our method

Predictions. 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 in 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 of 8 easy-to-compute handcrafted features, described below.

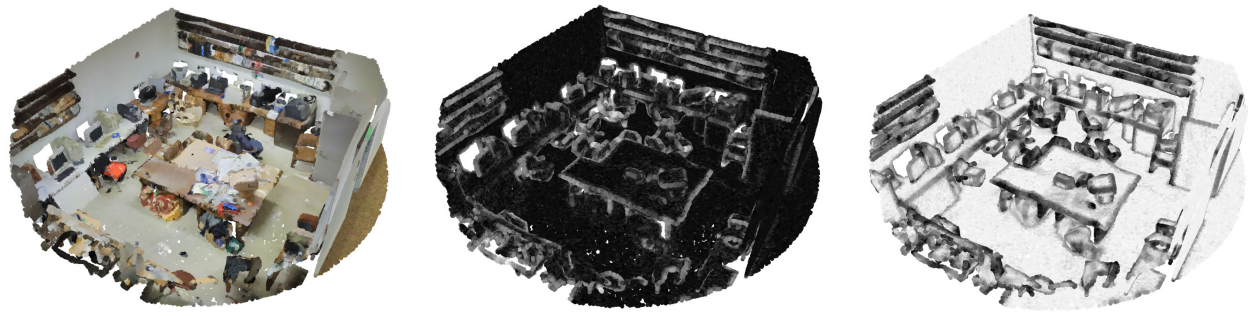
- *Radiometric features* (3 or 1): RGB colors are available for S3DIS and KITTI-360, and intensity values for DALES. These radiometric features are normalized to $[0, 1]$ at preprocessing time. For KITTI-360, we find that using the HSV color model yields better results.
- *Geometric features* (5): We use PCA-based features: *linearity*, *planarity*, *scattering*, [5] and *verticality* [7], 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 *elevation* as the distance between a point and the ground below it. Since the ground is neither necessarily flat nor horizontal, we use the RANSAC algorithm [6] 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}.

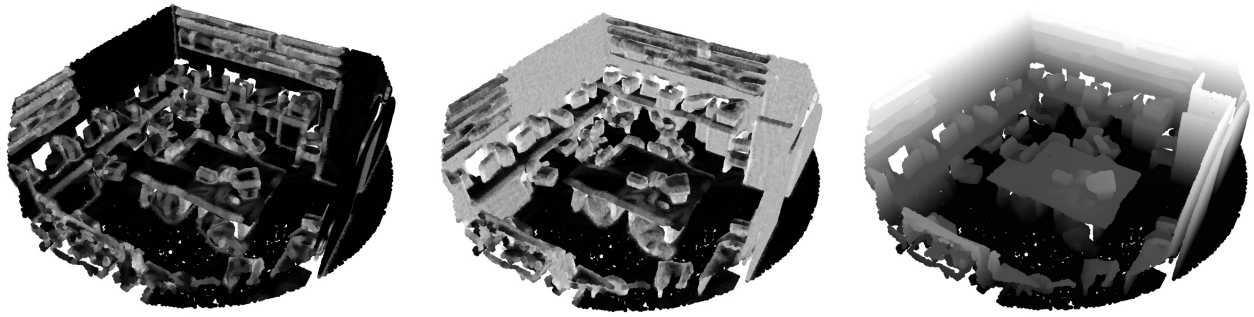
We provide an illustration of the geometric point features in Figure A-2, to help the reader apprehend these simple geometric descriptors.

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 [13] uses the Delaunay triangulation of the entire point cloud to compute such interfaces, while we propose a faster heuristic approach in Section A-6 called the *Approximate Superpoint Gap algorithm*. 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 [13], we characterize each pair of adjacent superpoints with the ratio of their *lengths*, *surfaces*, *volumes*, and *point counts*.
- *Pose features (7)*: For each superpoint, we define a normal vector as its principal component with the smallest eigenvalue. We then 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



(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.

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 $D_{\text{key}} + D_{\text{que}} + D_{\text{val}}$, for each level $i \geq 1$ of the encoder and the decoder.

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.

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	-	-
Normalized Room pos.	-0.7	-	-

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 [19, 3]. 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 [13] 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 [12] 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 by sorting them along the principal component of the selected 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 hierarchical multilevel partition of the point cloud into 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 [12]:

$$\mathcal{J}(e; f, \mathcal{G}, \lambda) := \|e - f\|^2 + \lambda \sum_{(u,v) \in \mathcal{E}} w_{u,v} \left[e_u \neq e_v\right] \ , \tag{A-1}$$

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

$\mathcal{J}$ 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 $\mathcal{P}$ and its associated variable $f^e \in \mathbb{R}^{D\times|\mathcal{P}^e|}$ such that e is constant within each segment p of $\mathcal{P}^e$ with value f^e_p. The partition $\mathcal{P}^e$ also induces a graph $\hat{\mathcal{G}}^e := (\mathcal{P}^e, \mathcal{E}^e, w^e)$ 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:

$$\mathcal{E}^e := \{(U,V) \mid U,V \in \mathcal{P}^e, (U\times V)\cap\mathcal{E} \neq \varnothing\} \quad \text{(A-2)}$$

$$\text{For } (U,V)\in\mathcal{E}^e,\ w^e_{U,V} := \sum_{(u,v)\in U\times V\cap\mathcal{E}} w_{u,v} \quad \text{(A-3)}$$

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

$$f^e, \mathcal{P}^e, \hat{\mathcal{G}}^e := \text{partition}\,(e)\ . \quad \text{(A-4)}$$

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 nonnegative regularization strengths defining the coarseness of the successive partitions. In particular, We chose λ_1 such that $|\mathcal{P}_1|/|\mathcal{P}_0 \sim 30$ in our experiments.

We first define $\hat{\mathcal{G}}_0$ as the point-level adjacency graph $\hat{\mathcal{G}}$ and f_0 as f. We can now define the levels of a hierarchical partition $\mathcal{P}_i$ for $i \in [1, I]$:

$$f_i, \mathcal{P}_i, \hat{\mathcal{G}}_i := \text{partition}\Big(\underset{e\in\mathbb{R}^{D\times|\mathcal{P}_{i-1}|}}{\arg\min}\ \mathcal{J}\left(e; f_{i-1}, \hat{\mathcal{G}}_{i-1}, \lambda_{i-1}\right)\Big). \quad \text{(A-5)}$$

Given that the optimization problems defined in Eq. (A-5) for $i > 1$ operate on the component graphs $\hat{\mathcal{G}}_i$, which are smaller than $\hat{\mathcal{G}}_0$, the first partition is the most demanding in terms of computation.

Note that we used the hat notation $\hat{\mathcal{G}}_i$, because these graphs are only used for computing the hierarchical partitions $\mathcal{P}_i$, and should be distinguished from the the superpoint graphs $\mathcal{G}_i$ on which is based our self-attention mechanism, constructed from $\mathcal{P}_i$ as explained in Section A-6.

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 coordinates with a normalization factor μ in m^{-1} to limit the size of the superpoints. We recommend setting 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 partitions depends on the regularization strength λ as defined in Section **??**. Finer partitions should generally lead to better results but to an increase in training time and memory requirement. We chose a ratio $|\mathcal{P}_0|/|\mathcal{P}_1| \sim 30$ 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.

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}	$[D_{\text{val}} + D_{\text{val}} + D^{\text{pos}}_{\text{point}}, D_{\text{val}}, D_{\text{val}}]$

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 [2] 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 $D^{\text{geof}}_{\text{point}} = 4$ 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 models evaluated 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 not 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 example, 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 behavior 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: **Impact of 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: **Impact of 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: **Impact of 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 introduced in Section 3.3 : predicting the most frequent label for $\mathcal{P}_1$ and the distribution for $\mathcal{P}_2$. We use "freq-$\mathcal{P}_i$" to refer to the prediction of the most frequent label applied the $\mathcal{P}_i$ partition. Similarly, "dist-$\mathcal{P}_i$" denotes the prediction of the distribution of labels within each superpoint of the partition $\mathcal{P}_i$.

We observe a consistent improvement across all datasets by adding the dist-$\mathcal{P}_i$ supervision. This illustrates the benefits of supervising higher-level partitions, despite their lower purity. Moreover, supervising $\mathcal{P}_1$ with the distribution rather than the most frequent label leads to a further performance drop. This validates our choice to consider $\mathcal{P}_1$ superpoints as 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
freq-$\mathcal{P}_i$-$\mathcal{P}_1$ dist-$\mathcal{P}_i$-$\mathcal{P}_2$	**76.0**	**63.5**	**79.6**
freq-$\mathcal{P}_1$	-0.2	-0.8	-0.8
dist-$\mathcal{P}_i$-$\mathcal{P}_1$	-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 S3DIS, which encompasses 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 [15]	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 [13]	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 [4]	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 [11]	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 [19]	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.[20]	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 [18]	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 [10]	**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 [15]	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 [13]	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 [1]	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 [4, 18]	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 [11]	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 [19]	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 [18]	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 [4, 18]	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 [18]	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++ [16]	68.3	94.1	91.2	75.4	30.3	79.9	46.2	40.0	89.1
ConvPoint [1]	67.4	96.9	91.9	75.5	21.7	86.7	29.6	40.3	96.3
SPG [13]	60.6	94.7	87.9	62.9	18.7	65.2	33.6	28.5	93.4
PointCNN [14]	58.4	**97.5**	91.7	40.6	40.8	26.7	52.6	57.6	95.7
KPConv [19]	**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

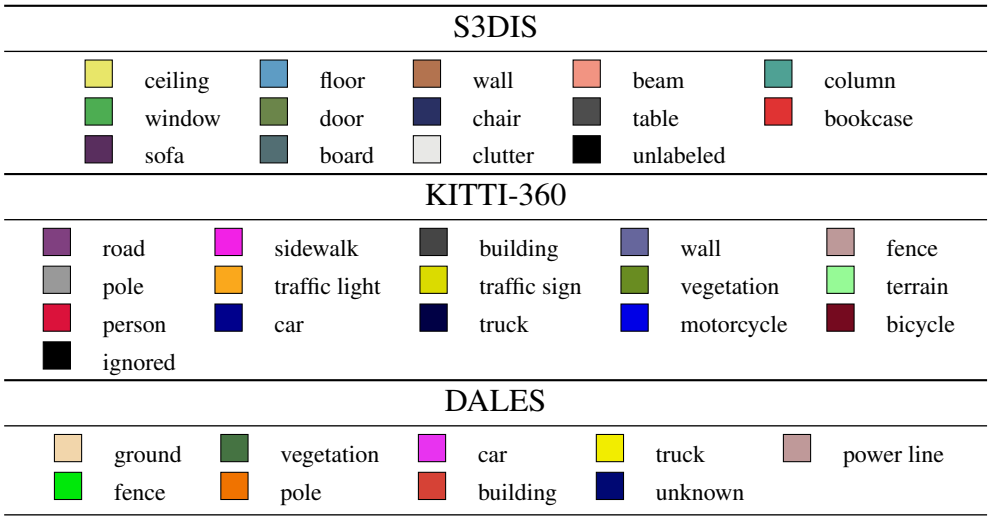


Figure A-3: **Colormaps.**

References

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

[2] 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.

[3] 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.

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

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

[6] 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.

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

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

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

[10] 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.

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

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

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

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

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

[16] 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.

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

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

[19] 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.

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