Point Cloud Oversegmentation with Graph-Structured Deep Metric Learning

Loic Landrieu, Mohamed Boussaha

Point Cloud Oversegmentation with Graph-Structured Deep Metric Learning

Loic Landrieu[1], Mohamed Boussaha[2]
Univ. Paris-Est, IGN-ENSG, LaSTIG, [1]STRUDEL, [2]ACTE, Saint-Mandé, France
loic.landrieu@ign.fr,mohamed.boussaha@ign.fr

Abstract

We propose a new supervized learning framework for oversegmenting 3D point clouds into superpoints. We cast this problem as learning deep embeddings of the local geometry and radiometry of 3D points, such that the border of objects presents high contrasts. The embeddings are computed using a lightweight neural network operating on the points' local neighborhood. Finally, we formulate point cloud oversegmentation as a graph partition problem with respect to the learned embeddings.

This new approach allows us to set a new state-of-the-art in point cloud oversegmentation by a significant margin, on a dense indoor dataset (S3DIS) and a sparse outdoor one (vKITTI). Our best solution requires over five times fewer superpoints to reach similar performance than previously published methods on S3DIS. Furthermore, we show that our framework can be used to improve superpoint-based semantic segmentation algorithms, setting a new state-of-the-art for this task as well.

1. Introduction

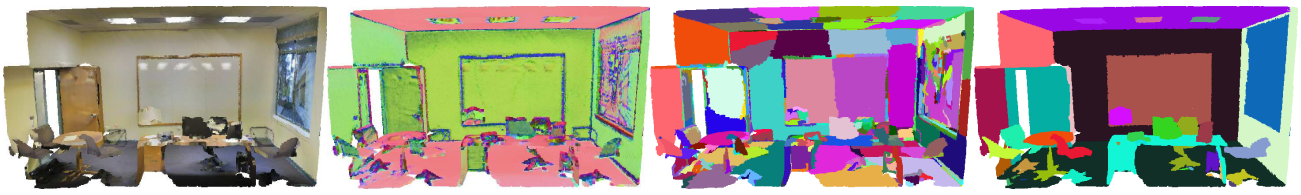
The interest of segmenting point clouds into sets of points known as superpoints—the 3D equivalent of superpixels— as a preprocessing step to their analysis has been extensively demonstrated [27, 39, 35, 7, 50]. However, these unsupervized methods rely on the assumption that segments which are geometrically and/or radiometrically homogeneous are also semantically homogeneous. This assertion should be challenged, especially since the quality of any further analysis is limited by the quality of the initial oversegmentation. Our objective in this paper is to formulate a supervized framework for oversegmentating 3D point clouds into semantically pure superpoints in order to facilitate their semantic segmentation.

Although superpixel-based methods and deep learning have both been around for a long time in computer vision, convolutional neural networks have only recently been used for superpixel oversegmentation. Notably, [32] introduced a loss function emulating oversegmentation metrics, and which is compatible with graph-based clustering methods. [24] propose a fully differentiable version of the SLIC superpixel algorithm [1], allowing for end-to-end training of spatial clustering methods. Both approaches have shown promising results, displaying significant improvement upon methods relying on handcrafted descriptors. In this paper, we build upon these ideas, albeit in the 3D setting.

We propose formulating point cloud oversegmentation as a deep metric learning problem structured by an adjacency graph defined on an input 3D point cloud. We introduce the *graph-structured contrastive loss*, a loss function which learns to embed 3D points homogeneously within objects and with high contrast at their interface. This loss can be adapted to the non-differentiable task of oversegmentation by using our *cross-partition weighting* strategy. The points' embeddings themselves are computed from the points' local geometry and radiometry by a lightweight model inspired from PointNet [36] and called *Local Point Embedder* (LPE). Finally, the superpoints are defined as a piecewise-constant approximation of the learned embedding in the adjacency graph, in the manner of [17].

Furthermore, we define the end-goal of our point cloud oversegmentation as assisting semantic segmentation methods by providing semantically pure superpoints. We show that our approach can be integrated with the superpoint graph approach of [27] to significantly improve the partition step, and consequently the resulting semantic segmentation. The contributions of this paper are as follows:

- We present the first supervized framework for 3D point cloud oversegmentation;
- We introduce the graph-structured contrastive loss, which can be combined with our cross-partition weighting strategy to produce point embeddings with high contrast at objects' borders;
- We introduce the local point embedder, a lightweight architecture, inspired by [36], to embed the local geometry and radiometry of 3D points in a compact way;
- We significantly improve the state-of-the-art of point cloud oversegmentation for two well-known and very different datasets;



(a) Input Point Cloud (b) Learned Embedding (c) Oversegmentation (d) True Objects

Figure 1: Illustration of our framework on a hard-to-segment scene with a white board on a white wall: a colored point cloud is given as input (a), an embedding is computed for each point (b), which allows a clustering technique to compute an oversegmentation (c), which closely follows the ground truth (d). Throughout the figures of this paper, the embeddings are projected into a 3-dimensional space to allow color visualization.

- When combined with the superpoint graph semantic segmentation method, our approach improves upon the state-of-the-art for this task as well.

2. Related work

Superpixels/ Supervoxels: There is a large body of literature on the oversegmentation of images into superpixels [44] and videos into supervoxels [51]. These methods can be divided into two groups: graph-based, which exploit the pixels' connectivity [11, 16, 31], and cluster-based, which use the pixels' relative positions [1, 46, 52, 28]. Recently, deep learning methods have been successfully used to develop supervized superpixels oversegmentation approaches, either graph-based [32], or cluster-based [24].

Oversegmentation of 3D Point Clouds: The aforementioned methods perform well on images, but rely on the regular structure of pixels. 3D point clouds, as unordered point sets with irregular distributions, require special attention. [4] propose three extensions of 2D local variation graph-based method [11] to 3D oversegmentation and study different strategies for constructing the graph, edge weights, and subgraph merging. [43] introduce a graph-structured approach which exploits the structure of LiDAR sensors to remove edges corresponding to boundary points. [34] propose a cluster-based method based on the k-means algorithm and octrees. However, this method remains sensitive to the clusters' initialization. [12] use the visual saliency of RGBD images to initialize clustering. [30] propose a clustering method which does not require such initialization, and is therefore less sensitive to the irregular densities of LiDAR point clouds. Likewise, [17] introduce an initialization-free segmentation model formulated as a graph-structured optimization problem. All these methods rely on hand-crafted geometric and/or colorimetric features.

Deep Learning for 3D Point Clouds: The work in [36] has pioneered the use of deep learning for 3D point cloud processing. However, this usage has so far only been used for semantic segmentation [29, 45, 9, 41, 38, 37, 53, 49], object detection [56], or reconstruction [15]. To the best of our knowledge, no supervised 3D point oversegmentation technique that leverages deep learning-based embeddings to generate superpoints has been developed yet.

Metric Learning: Metric learning aims to learn a similarity function between data points with properties corresponding to a given task [25]. In practice, an embedding function associates each data point with a feature vector attuned to a given objective. These objectives can be related to classification [13, 40], or clustering [42, 19], among many other applications (see [2] for a useful taxonomy). In the context of deep learning, this can be achieved by using a well-chosen loss, such as the contrastive loss [8, 5]; the triplet loss [20] or some of its variants [48]. Notably, metric learning has recently been used to improve the quality of learned features for a 3D point semantic segmentation task [10]. However, our task is different in the sense that our embeddings are related to oversegmentation through a graph partition problem rather than classification.

3. Method

Our goal is to produce a high-quality 3D-point cloud oversegmentation, so that it can be in turn used by superpoint-based semantic segmentation algorithms. This translates into the following three properties:

($\mathcal{P}1$) **object-purity**: superpoints must not overlap over objects, especially if their semantics are different;

($\mathcal{P}2$) **border recall**: the interface between superpoints must coincide with the borders between objects;

($\mathcal{P}3$) **regularity**: the shape and contours of the superpoints must be simple.

Our approach can be broken down into two steps: in Section 3.1 we present the local cloud embedder, a simple neural network which associates each point with a compact embedding that captures its local geometry and radiometry. In Section 3.2, we describe how we compute a point cloud oversegmentation from this embedding using either graph or cluster-based oversegmentation algorithms.

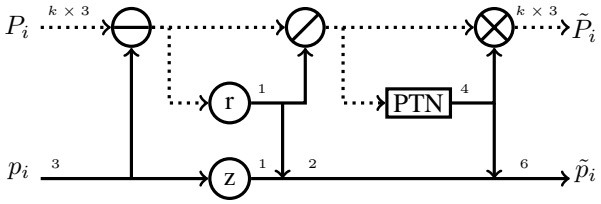


Figure 2: Architecture of the spatial transform network. It takes a point's coordinate as point-input p_i and the coordinates of its neighbors as set-input P_i. The vertex r computes the radius of a point cloud (1), the vertex z extract the vertical coordinate of a point's position, and the vertex PTN is a small PointNet-like network (2) which outputs a 2×2 rotation matrix around the z axis (4). In this and subsequent figures, set-features (respectively point-features) are represented by a dotted line (respectively a solid line). The numbers above the lines represent the size of the channels.

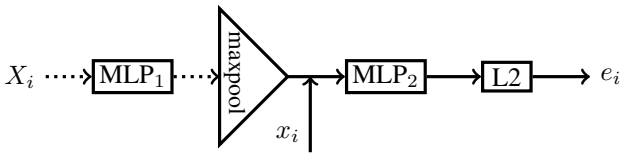


Figure 3: Architecture of the local point embedder (LPE) (7), which computes an embedding set-feature X_i and point-feature x_i encoding the local radiometry and the normalized geometry. The L_2 block normalizes the output on the unit sphere (6).

Throughout this paper we will stress the difference between *set-features*, which are unordered sets of descriptors (such as information related to the neighbors of a point), and *point-features*, which characterize a specific point. Set features will always be capitalized, while point-features will use lowercase.

Let us consider a point cloud C, with each point i defined with its position $p_i \in \mathbb{R}^3$ and d-dimensional radiometric information $r_i \in \mathbb{R}^d$ (this can be colors if available, or intensity for LiDAR scans, or be ignored if none is available). Each point i is associated with the set-features P_i and R_i, respectively comprised of the position and radiometry of its k nearest neighbors N_i in the input cloud: $P_i = \{p_j \mid j \in N_i\}, R_i = \{r_j \mid j \in N_i\}$. For ease of notation, any operator or function f applied to a set-feature X is to be understood as being applied to all its elements: $f(X) = \{f(x) \mid x \in X\}$.

3.1. Local Point Embedding

Our objective is to associate to each point a compact m-dimensional embedding e_i characterizing its point-features (position, color, etc.) and the geometry and radiometry of its local neighborhood. The embeddings are constrained to be within the m-unit sphere $\mathbb{S}_m$, as suggested by [47], to prevent collapse during the training phase, and to normalize their distance with one another.

To this end, we introduce the Local Point Embedder (LPE), a lightweight network inspired by PointNet [36]. However, unlike PointNet, LPE does not try to extract information from the whole input point cloud, but rather encodes each point based on purely local information. Here, we describe the different units of our network.

Spatial Transform: This unit takes the positions of a target point p_i and its local k-neighborhood P_i, as represented in Figure 2. It normalizes the neighbors' coordinates around p_i, and such that the standard deviation of the point's position is equal to 1 (3). Then, this neighborhood is rotated around the z axis with a 2×2 rotation matrix computed by small PointNet network PTN (4). As advocated by [23], these steps aim to standardize the position of the neighborhood clouds of each point. This helps the next network to learn position distribution. Along the normalized neighborhood position $\tilde{P}_i$, this unit also outputs geometric point-features $\tilde{p}_i$ describing the elevation $p_i^{(z)}$, the neighborhood radius, as well as its original orientation (through the 4 values of the rotation matrix: $[\Omega_{x,x}, \Omega_{x,y}, \Omega_{y,x}, \Omega_{y,y}]$)(5). By keeping track of the normalization operations, the embedding can stay covariant with the original neighborhood's radius, height, and original orientation, even though the points' positions have been normalized and rotated.

$$(1)\ \text{rad} = \text{std}(P_i) \qquad \tilde{P}_i = \{p \times \Omega \mid p \in P'_i\} \quad (4)$$

$$(2)\ \Omega = \text{PTN}(\tilde{P}_i) \qquad \tilde{p}_i = [p_i^{(z)}, \text{rad}, \Omega] \quad (5)$$

$$(3)\ P'_i = (P_i - p_i)/\text{rad}$$

Local Point Embedder: The LPE network, represented in Figure 3, computes a normalized embedding from two inputs: a point-feature x_i and a set-feature X_i. As in PointNet [36], the set-features are first processed independently by a multi-layer perceptron (denoted MLP_1) comprised of a succession of layers in the following order: linear, activation (ReLu [33]), normalization (batch [22]), and so on. The resulting set-features are then maxpooled into a point-feature, which is concatenated with the input point-feature. The resulting vector is processed through another multi-layer perceptron MLP_2 (7), and finally normalized on the unit sphere.

The embeddings e_i are computed for each point i of C through a shared LPE (8). The input set-feature X_i is set as the concatenation of the neighbour's transformed position $\tilde{P}_i$ and their radiometric information R_i, while the input point-feature x_i is composed of the neighborhood geometric point-feature $\tilde{p}_i$ and the radiometry r_i of point i.

$$\text{L}_2(\cdot) = \cdot / \|\cdot\| \quad (6)$$

$$\text{LPE}(X_i, x_i) = \text{L}_2\left(\text{MLP}_2\left(\left[\max\left(\text{MLP}_1(X_i)\right), x_i\right]\right)\right) \quad (7)$$

$$e_i = \text{LPE}([\tilde{P}_i, R_i], [\tilde{p}_i, r_i]) \quad (8)$$

3.2. Graph-Based Point Cloud Oversegmentation

3.2.1 The Generalized Minimal Partition Problem

Once the embeddings are computed, we define the superpoints with respect to an adjacency graph $G = (C, E)$ derived from the point cloud C. Note that E can be obtained from the neighbors' structure used for the LPE. However, we find that much smaller neighborhoods are needed to capture the cloud's adjacency structure than to describe the local neighborhood of points. As proposed by [17], we define the superpoints as the constant connected components in G of a piecewise-constant approximation of the embeddings $e \in \mathbb{S}_m^C$. This approximation is the solution $f^\star$ of the following optimization problem:

$$f^\star = \underset{f \in \mathbb{R}^{C \times m}}{\arg\min} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} \left[f_i \neq f_j\right], \quad (9)$$

with $w \in \mathbb{R}_+^E$ the edges' weight and $[x \neq y]$ equal to 0 if $x = y$ and 1 otherwise. To encourage the network to split along high contrast areas, we define the edge weight as $w_{i,j} = \lambda \exp\left(\frac{-1}{\sigma}\|e_i - e_j\|^2\right)$, with parameters $\lambda, \sigma \in \mathbb{R}^+$.

Problem (9), known as the *generalized minimal partition* (GMP) and introduced by [26], is neither continuous, differentiable, nor convex, and therefore the global minimum cannot be realistically retrieved. However, the ℓ_0-cut pursuit algorithm [26] allows for fast approximate solutions.

The contour penalty automatically implements ($\mathcal{P}3$) for reasonable parameterization of the problem. Note that the optimization variable f can take its values in $\mathbb{R}^{C \times m}$, while each embedding e_i is constrained on the m-sphere. This is a limitation of our approach due to efficiency concerns. It can lead to some suboptimal approximate solutions. However, we show in the numerical experiments that the learned embeddings lead to satisfactory partitions.

3.2.2 Graph-Structured Contrastive Loss

As mentioned earlier, the semantic purity property ($\mathcal{P}1$) is the first quality of superpoints. Once could imagine taking a metric estimating the semantic purity of the solution of (9) as a loss function. However, the GMP is a non-continuous non-convex optimization problem, and computing connected components on a graph is inherently non-differentiable. This makes optimizing directly with respect to properties of the partition very hard, if not impossible.

Instead, we note that if the border recall property ($\mathcal{P}2$) is implemented (*i.e.* superpoints and objects share the same boundaries), then ($\mathcal{P}1$) ensues. Therefore, we propose a surrogate loss called *graph-structured contrastive loss* focusing on correctly detecting the borders between objects. To this end, we define E_{intra} (resp. E_{inter}) the set of *intra-edges* (resp. *inter-edges*) as the set of edges of G between points within the same object (resp. point from different adjacent objects).

In the spirit of the original contrastive loss [8], our loss encourages embeddings of vertices linked by an intra-edge to be similar, while rewarding different embeddings when linked by an inter-edge:

$$\ell(e) = \frac{1}{|E|}\left(\sum_{(i,j) \in E_{\text{intra}}} \phi\left(e_i - e_j\right) + \sum_{(i,j) \in E_{\text{inter}}} \mu_{i,j} \psi\left(e_i - e_j\right)\right),$$

with ϕ (resp. ψ) a function minimal (resp. maximal) at 0, and $\mu_{i,j} \in \mathbb{R}^{E_{\text{inter}}}$ a weight on inter-edges. A point embedding function minimizing this loss will be uniform within objects and have stark contrasts at their interface. Consequently, the components of the piece-wise constant approximation of (9) should follow the objects' borders. This loss differs from the triplet loss [20, 47], as it involves all vertices within a graph (or a sub-graph) at once, and not just an anchor and related positive/negative examples. In this way, it bypasses the problem of example picking altogether. Indeed, the positive and negative examples are directly given by the adjacency structure set by E_{intra} and E_{inter}. It differs from [10] as it does not try to learn semantic information, but rather to compute a signal on a graph such that its constant approximation respects certain properties, with no attention to semantics. Indeed, objects of different classes can share the same embeddings as long as they are never adjacent, such as floors and ceilings for indoor scenes.

We chose ϕ, the function promoting intra-object homogeneity as $\phi(x) = \delta(\sqrt{\|x\|^2/\delta^2 + 1} - 1)$ with $\delta = 0.3$ (represented in Figure 4). This means that the first term of ℓ is the (pseudo)-Huber graph-total variation on the E_{intra} edge [21, 6], promoting smooth homogeneity of embeddings within the same object.

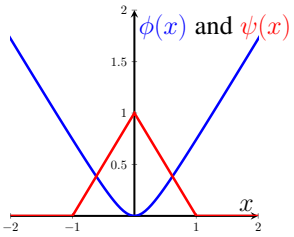


Figure 4: The functions ϕ (in blue) and ψ (in red) used in the graph-structured contrastive loss.

With $\psi(x) = \max(1 - \|x\|, 0)$, the second part of ℓ is the opposite of the truncated graph-total variation [55] on the inter-edges. It penalizes similar embeddings at the border between objects. Conscious that our embeddings are restricted to the unit sphere, we threshold this function for differences larger than 1 (corresponding to a 60 degree angle). In other words, $\psi(x)$ encourages vertices linked by an inter-edge to take embeddings with an euclidean distance of 1, but does not push for a larger difference.

Note that any embeddings that are constant within ob-

jects, and with a difference of at least 1 between adjacent objects, will have 0 loss. The four-color theorem [14] tells us that it is always possible as long as the dimension of our embedding is at least 3. However, because embeddings are computed by the LPE, borders which do not present recognizable geometric or radiometric configurations cannot be recovered by our method.

3.2.3 Cross-Partition Weighting

The choice of $\mu_{i,j}$ plays a crucial role in the efficiency of the graph-structured contrastive loss. Although ($\mathcal{P}2$) does imply ($\mathcal{P}1$), small errors in the former can have drastic consequences in the latter. Indeed, a single missed edge can erroneously fuse two large superpoints covering different objects. Therefore, we need to incorporate the induced partition's purity into the loss.

[32] introduced the segmentation-aware affinity loss (SEAL) implementing this idea. They propose weighting intra-edges as 1, and inter-edges as $\mu_{i,j} = 1 + |S| - |S \backslash O_S|$ for i and j within the same superpoint S, with O_S the *majority-object*, *i.e.* the object for which most points of S belongs to. Although [32] boasts impressive results for superpixel oversegmentation, we were not able to extend this success within our framework. We believe this stems from three reasons: (i) all border edges of a superpoint are weighted identically regardless of their influence on the purity and the size of the interface; (ii) as soon as a superpoint no longer overlaps an object's border, its weight decreases dramatically to 1, making the loss very unstable; (iii) [32] uses a different graph-based clustering[31].

To overcome these limitations, we introduce the cross-partition weighting strategy. We first compute the *cross-segmentation graph* $\mathcal{G} = (\mathcal{C}, \mathcal{E})$, defined as the adjacency graph of the cross-partition $\mathcal{C}$ of C between the superpoints partition $\mathcal{S}$ and the object partition $\mathcal{O}$. In other words, $\mathcal{C}$ is the set of connected components of the graph G when all edges either between objects *or* between superpoints are removed, and the *super-edge* (*i.e.* set of edges) $(U, V) \in \mathcal{E}$ is the set of inter-edges of E_{inter} between U and V in $\mathcal{C}$:

$$\mathcal{C} = \{O \cap S \mid O \in \mathcal{O}, S \in \mathcal{S}\}$$
$$\mathcal{E} = \{\{(i,j) \in (U \times V) \cap E_{\text{inter}}\} \mid U, V \in \mathcal{C}\}.$$

We associate the following weight $\mu_{U,V}$ to each superedge (U, V) and $\mu_{i,j}$ to each edge:

$$\mu_{U,V} = \frac{\mu \min(|U|, |V|)}{|(U,V)|} \qquad \text{for } (U,V) \in \mathcal{E}$$
$$\mu_{i,j} = \mu_{U,V} \qquad \text{for all } (i,j) \in (U,V)$$

with μ a parameter of the model. Such weights simultaneously take into account the influence of the edges in the purity and the shape of the interfaces. Indeed, should an edge of the superedge (U, V) be missed as a border, the superpoints U and V would be merged. Since U and V cover different objects (by definition of $\mathcal{E}$), such a merger would induce at least $\min(|U|, |V|)$ vertices *trespassing*, *i.e.* not being in the majority-object of the merged superpoint.The weights are also divided by the number of edges constituting the interface between U and V in order to distribute evenly the penalty over the number of edges constituting an interface. This prevents long borders from being over-represented in the loss. See Figure 5 for an illustration.

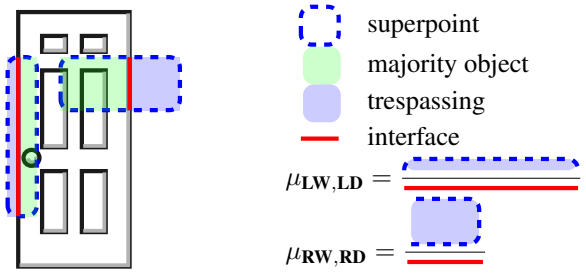


Figure 5: Illustration of the cross-partition weighting strategy on a scene comprised of a door (**D**) and a wall (**W**). Two superpoints **L** (left) and **R** (right) overlap the door. The superedge (**LW**, **LD**)(resp. (**RW**, **RD**)) represent the adjacency between the part of the left (resp. right) superpoint covering the wall and the part covering the door. With fewer trespassing points and a longer interface than (**RW**, **RD**), the weights of the edges constituting (**LW**, **LD**) are smaller.

3.3. Cluster-Based Oversegmentation

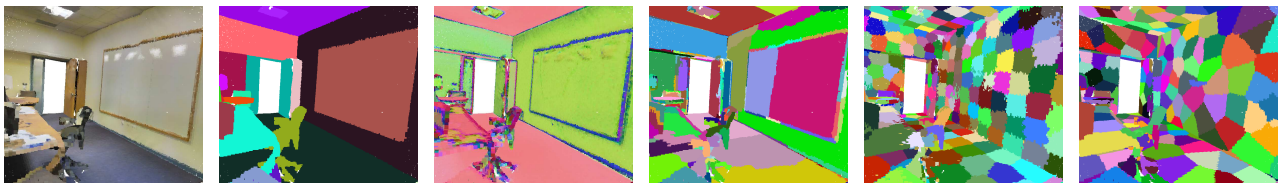
We also implemented a generalization of the method of [24] to the 3D setting. The main advantage of this approach is that the loss can directly implement ($\mathcal{P}1$) through the cross-entropy of the averaged semantic classes within superpoints. However, this approach remains hindered by its sensitivity to the superpoint initialization, and its inability to adapt the superpoints' size to the local complexity of the scene. Furthermore, as it bypasses ($\mathcal{P}3$), it produces superpoints with complicated contour.

3.4. Implementation Details

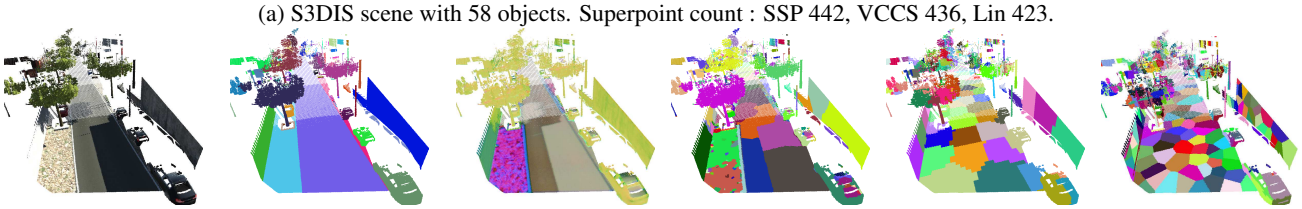
We use a modified version of the ℓ_0-cut pursuit algorithm[1] [26], with two main differences:

- to prevent the creation of many small superpoints in regions of high contrast, we merge components greedily with respect to the objective energy defined in (9), as long as they are smaller than a given threshold ;
- we heuristically improved the forward step (8) from [26], such that the regularization strength increases geometrically by a factor (of 0.7) along the iterations.

[1] https://github.com/loicland/cut-pursuit



(a) S3DIS scene with 58 objects. Superpoint count : SSP 442, VCCS 436, Lin 423.



Input cloud | Ground truth objects | LPE embeddings | SSP (ours) | VCCS [34] | Lin in [30]

(b) vKITTI scene with 233 objects. Superpoint count: SSP 420, VCCS 422, Lin 425.

Figure 6: Illustration of the oversegmentations of our framework, and from competing algorithms.

This helps improve the quality of the lower optima retrieved, and consequently the oversegmentation's.

To limit the size of the superpoints we concatenate to the points' embeddings their 3D coordinates in (9) multiplied by a parameter α_{spatial}, in the manner of [1]. This determines the maximum size that superpoints can reach.

In all our experiments, we set m the dimension of our embeddings to 4. We choose a light architecture for the LPE, with less than $15,000$ parameters. The exact network configurations for each dataset are detailed in the appendix.

4. Numerical Experiments

4.1. Datasets

We evaluate our approach on two datasets of different natures. The first one is S3DIS [3], composed of dense indoor scans of rooms in an office setting. The second one is vKITTI [9], an outdoor dataset of urban scenes that mimics sparse LiDAR acquisitions. Note that only S3DIS has individual object annotation. We consider the objects of vKITTI to be the connected components of the semantic labels in the adjacency graph G. For vKITTI, we consider the performance of our algorithm with and without color information. Both datasets are large scale (close to 600 million points for S3DIS and close to 15 million for vKITTI). We subsample them using a regular grid of voxels (3cm wide for S3DIS and 5cm wide for vKITTI). In each voxel, we average the position and color of the contained points. This allows us to decrease the computation time and memory load.

4.2. Point Cloud Oversegmentation

Evaluation Metrics: There are many standard metrics which assess the quality of point cloud oversegmentations with respect to properties ($\mathcal{P}1$), ($\mathcal{P}2$), and ($\mathcal{P}3$). In particular, the Boundary Recall (BR) and Precision (BP) are used to evaluate the ability of the superpoints to adhere to, and not cross, object boundaries (($\mathcal{P}2$), ($\mathcal{P}3$)). In the literature, these measures are defined with respect to *boundary pixels* [34] or points [30]. However, we argue that transition occurs *between* points and not *at* points for point clouds. Consequently, we define $E^{\text{pred}}_{\text{inter}}$ the set of predicted transition, *i.e.* the subset of edges of E that connect two points of C in two different superpoints. These metrics are often given with respect to a tolerance, *i.e.* the distance at which a predicted transition must take place from an actual object's border for the latter to be considered retrieved. We set this distance to 1 edge, which leads us to define $E^{(1)}_{\text{inter}}$ the set of inter-edges expanded to all directly adjacent edges in E:

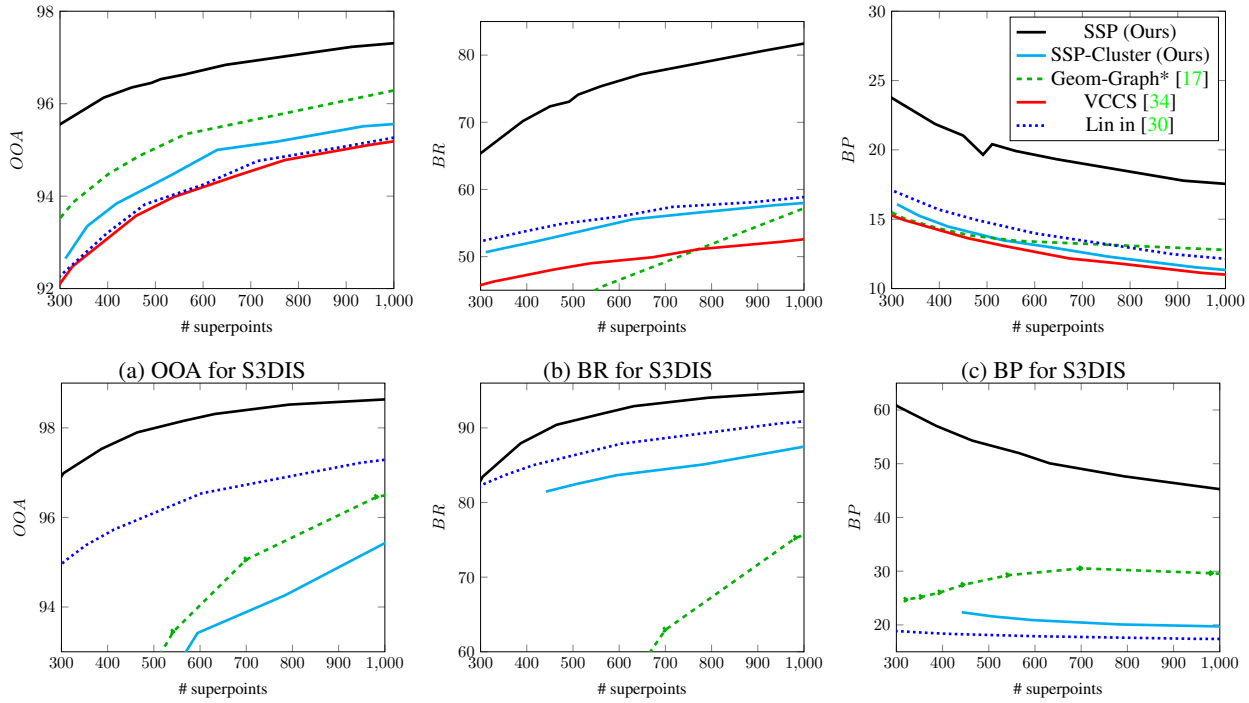
$$E^{(1)}_{\text{inter}} = \{(i,j) \in E \mid \exists (i,k) \text{ or } (j,k) \in E_{\text{inter}}\}\,.$$

This allows us to define the boundary recall and precision with 1 edge tolerance for a set of predicted transition $E^{\text{pred}}_{\text{inter}}$:

$$BR = \frac{\mid E^{\text{pred}}_{\text{inter}} \cap E^{(1)}_{\text{inter}} \mid}{\mid E_{\text{inter}} \mid},\ BP = \frac{\mid E^{\text{pred}}_{\text{inter}} \cap E^{(1)}_{\text{inter}} \mid}{\mid E^{\text{pred}}_{\text{inter}} \mid}.$$

Since the end-goal of our point cloud oversegmentation framework is to provide useful superpoints for semantic segmentation, we define the *Oracle Overall Accuracy* (OOA). To assess object purity ($\mathcal{P}1$), this metric characterizes the accuracy of the labeling that associates each superpoint S of a segmentation $\mathcal{S}$ with its majority ground-truth label. Formally, let $l \in \mathcal{K}^C$ be the semantic labels of each point within a set of classes $\mathcal{K}$, we define the OOA of a point cloud segmentation $\mathcal{S}$ as:

$$l^{\text{oracle}}(S) = \text{mode}\,\{l_i \mid i \in S\}$$
$$OOA = \frac{1}{\mid C \mid} \sum_{S \in \mathcal{S}} \sum_{i \in S} \left[l_i = l^{\text{oracle}}(S)\right],$$



(a) OOA for S3DIS (b) BR for S3DIS (c) BP for S3DIS

(d) OOA for vKITTI (e) BR for vKITTI (f) BP for vKITTI

Figure 7: Performance of the different algorithms on the 6-fold S3DIS dataset (a, b, c), and the 6-fold vKITTI dataset (d, e, f). The results of the method annotated with an asterix * have not been reported before. SSP-Cluster and VCCS are not represented for vKITTI for the sake of legibility as their performance is too low.

with $[x = y]$ the function equal to 1 if $x = y$ and 0 otherwise. Note that the OOA is closely related to the ASA [31], but consider the majority labels of all points within a superpixel rather than the label of the objects with most overlap. In this sense, it is a tighter upper bound to the achievable accuracy of a superpoint-based semantic classification algorithm using $\mathcal{S}$. This metric is also more fair than the undersegmentation error [28] for other methods such as [17], or our cluster-based approach, as they do not try to retrieve objects directly, but rather regions of C with homogeneous semantic labeling.

Competing algorithms: We denote by **SSP** (Supervized SuperPoint) our method when using LPE to learn point embeddings and then derive the superpoints using the graph-based methods described in Section 3.2.2, and **SSP-Cluster** when using the cluster-based method defined in Section 3.3 instead. We first assess the benefit of learning embeddings by comparing our results to those of [17], dubbed here **Geom-Graph**. This method computes superpoints by solving the generalized minimal partition problem as well, but with handcrafted geometric features in place of our learned embeddings. We illustrate in Figure 7 the oversegmentations produced by our approach and two state-of-the-art algorithms: **VCCS** [34] and the work of Lin in [30].

We observe that our approach significantly outperforms the other approaches on all metrics. In particular, we remark that **SSP** only requires under 350 superpoints to reach a performance comparable with **VCCS** with over $1,800$ superpoints on S3DIS. Furthermore, the quality of the border is unmatched in our range of superpoints. The improvement is less significant on vKITTI, which could be due to the difficulty of constructing an adjacency graph on such a sparse acquisition. The performance is degraded further without color information, as some transition are not predictable with purely from the geometry. **Geom-Graph** performs well on the accuracy, but not on the boundary. This is expected as the handcrafted geometric features cannot detect some borders, such as adjacent walls. **SSP-Cluster** performs better than the unsupervized cluster-based method of Lin *et al.* , but still suffer from the typical limitations of clustering methods, such as sensitivity to initialization.

In terms of computational speed, the embeddings can be computed very efficiently in parallel on a GPU with over 3 million embeddings per second on a 1080Ti GPU. The bottleneck remains solving the graph partition problem in (9), which can process around $100,000$ points per second.

4.3. Semantic Segmentation

In Table 1 and Table 2, we show how our point cloud oversegmentation framework can be successfully used by

the superpoint-based semantic segmentation technique of [27][2] (**SPG**). We replace the unsupervized superpoint computation with our best-performing approach, **SSP**. We evaluate the resulting semantic segmentation using standard classification metrics: overall accuracy (OA), mean per-class accuracy (mAcc) and mean per-class intersection-over-union (mIOU). We observe a significant increase in the performance of **SPG**, beating concurrent methods on both datasets. In particular, we observe that our method allows for better retrieval of small objects (see detailed IoU in the appendix), which translates into much better per-class metrics, although the overall accuracy is not necessarily better than the latest state-of-the-art algorithms.

4.4. Ablation Study

In Table 3, we present an ablation study to empirically justify some of our design choices. To make things more legible, we present the increase/decrease of the 3 performance metrics at 500 superpoints (linearly interpolated) of alternative methods compared to ours, on the first cross-validation fold of the S3DIS dataset. In particular we present **Prop-weight**, an alternative version in which the cross-partition weighting is replaced by a simple inversely-proportional weighting of the inter/intra edges. Predictably, this method gives lesser results as the edges are not weighted according to their influence in the partition. However, since the weights of the intra-edge are proportionally higher, the border precision is improved. We implemented the weights of the segmentation-aware affinity loss of [32] as well for method **SEAL-weights**, with comparable results to the **Prop-weight**. In **+TV-TV**, we replace our choice of function ϕ and ψ in the loss by respectively $|\cdot|$ and $-|\cdot|$, so that our loss is closer to the pairwise affinity loss used by [10] (but still structured by the graph). However, this approach wouldn't give meaningful partition as the intra-edge term conflicts with the constraint that the embeddings are constrained on the sphere. Removing this restriction leads the collapse of the embeddings around 0. We also tried to stack the LPE in layers, using or not a residual structure comparable to the one used in [18] to increase their receptive fields (more details are given in the appendix). The best results were achieved with two layers: **2-Layers** and **2-Residuals**. However, we observe that when compared with LPE of a similar number of parameters, the gains are insignificant if not null. We conclude that to embed points in order to detect borders, a small receptive field with a shallow architecture is sufficient.

5. Conclusion

In this paper, we presented the first supervized 3D point cloud oversegmentation framework. Using a simple point embedding network and a new graph-structured loss function, we were able to achieve significant improvements compared to the state-of-the-art of point cloud oversegmentation. When combined with a superpoint-based semantic segmentation method, our method sets a new state-of-the-art of semantic segmentation as well. A video illustration is accessible at https://youtu.be/bKxU03tjLJ4. The source code will be made available to the community as well as trained networks in an update to the superpoint-graph repository[2]. Future work will focus on improving the solving method for the generalized minimum minimal partition problem to better handle spherically-bounded variables, and to improve its computational performance.

Method	OA	mAcc	mIoU
6-fold cross validation			
PointNet [36] in [9]	78.5	66.2	47.6
Engelmann *et al.* in [9]	81.1	66.4	49.7
PointNet++ [37]	81.0	67.1	54.5
Engelmann *et al.* in [10]	84.0	67.8	58.3
SPG [27]	85.5	73.0	62.1
PointCNN [29]	**88.1**	75.6	65.4
SSP + SPG (ours)	87.9	**78.3**	**68.4**
Fold 5			
PointNet [36] in [10]	-	49.0	41.1
Engelmann *et al.* in [10]	84.2	61.8	52.2
pointCNN [29]	85.9	63.9	57.3
SPG [27]	86.4	66.5	58.0
PCCN [49]	-	67.0	58.3
SSP + SPG (ours)	**87.9**	**68.2**	**61.7**

Table 1: Performance of different methods for the semantic segmentation task on the S3DIS dataset. The top table is for the 6-fold cross validation, the bottom table on the fifth fold only.

Method	OA	mAcc	mIoU
PointNet [36]	79.7	47.0	34.4
Engelmann *et al.* in [10]	79.7	57.6	35.6
Engelmann *et al.* in [9]	80.6	49.7	36.2
3P-RNN [54]	**87.8**	54.1	41.6
SSP + SPG (ours)	84.3	**67.3**	**52.0**

Table 2: Performance of different methods for the semantic segmentation task on the vKITTI dataset with 6-fold cross validation.

Method	# parameters	OOA	BR	BP
Best	13,816	96.2	73.3	22.1
Prop-weights	13,816	-2.6	-12.2	+10.4
SEAL-weights	13,816	-1.3	-11.3	+3.8
2-Layers	14,688	-0.1	-0.7	-0.3
2-Residuals	14,688	+0.0	-0.2	-0.7

Table 3: Impact of some of our design choice on S3DIS. **Best** is the **SSP** method with cross-partition weights.

[2] https://github.com/loicland/superpoint-graph

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Süsstrunk, et al. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 34(11), 2012. 1, 2, 6

[2] E. Aljalbout, V. Golkov, Y. Siddiqui, and D. Cremers. Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*, 2018. 2

[3] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. K. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016. 6

[4] Y. Ben-Shabat, T. Avraham, M. Lindenbaum, and A. Fischer. Graph based over-segmentation methods for 3d point clouds. *Computer Vision and Image Understanding*, 2018. 2

[5] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, 1994. 2

[6] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on Image Processing*, 6(2), 1997. 4

[7] J. Chen and B. Chen. Architectural modeling from sparsely scanned range data. *International Journal of Computer Vision*, 78(2-3), 2008. 1

[8] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, volume 1. IEEE, 2005. 2, 4

[9] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe. Exploring spatial context for 3d semantic segmentation of point clouds. In *ICCV Workshops*, 2017. 2, 6, 8

[10] F. Engelmann, T. Kontogianni, J. Schult, and B. Leibe. Know what your neighbors do: 3d semantic segmentation of point clouds. In *GMDL Workshop, ECCV*, 2018. 2, 4, 8

[11] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2), 2004. 2

[12] G. Gao, M. Lauri, J. Zhang, and S. Frintrop. Saliency-guided adaptive seeding for supervoxel segmentation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2017. 2

[13] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, 2005. 2

[14] G. Gonthier. Formal proof–the four-color theorem. *Notices of the AMS*, 55(11), 2008. 5

[15] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. Atlasnet: A papier-maché approach to learning 3d surface generation. In *CVPR*. IEEE, 2017. 2

[16] M. Grundmann, V. Kwatra, M. Han, and I. A. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, 2010. 2

[17] S. Guinard and L. Landrieu. Weakly supervised segmentation-aided classification of urban scenes from 3d lidar point clouds. In *ISPRS Workshop*, 2017. 1, 2, 4, 7

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 8

[19] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe. Deep clustering: Discriminative embeddings for segmentation and separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE*. IEEE, 2016. 2

[20] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015. 2, 4

[21] P. J. Huber et al. Robust regression: asymptotics, conjectures and monte carlo. *The Annals of Statistics*, 1(5), 1973. 4

[22] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 3

[23] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, 2015. 3

[24] V. Jampani, D. Sun, M. Liu, M. Yang, and J. Kautz. Superpixel sampling networks. In *ECCV*, 2018. 1, 2, 5

[25] B. Kulis et al. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4), 2013. 2

[26] L. Landrieu and G. Obozinski. Cut pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs. *SIAM Journal on Imaging Sciences*, 10(4), 2017. 4, 5

[27] L. Landrieu and M. Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*. IEEE, 2018. 1, 8

[28] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12), 2009. 2, 7

[29] Y. Li, R. Bu, M. Sun, and B. Chen. PointCNN. *arXiv preprint arXiv:1801.07791*, 2018. 2, 8

[30] Y. Lin, C. Wang, D. Zhai, W. Li, and J. Li. Toward better boundary preserved supervoxel segmentation for 3d point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 143, 2018. 2, 6, 7

[31] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy rate superpixel segmentation. In *CVPR*. IEEE, 2011. 2, 5, 7

[32] W.-C. T. M.-Y. Liu, V. J. D. S. Shao-Yi, C. M.-H. Yang, and J. Kautz. Learning superpixels with segmentation-aware affinity loss. In *CVPR*. IEEE, 2018. 1, 2, 5, 8

[33] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 3

[34] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter. Voxel cloud connectivity segmentation - supervoxels for point clouds. In *CVPR*, 2013. 2, 6, 7

[35] S. Pu, G. Vosselman, et al. Automatic extraction of building features from terrestrial laser scanning. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5), 2006. 1

[36] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CVPR, IEEE*, 1(2), 2017. 1, 2, 3, 8

[37] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 2, 8

[38] G. Riegler, A. O. Ulusoy, and A. Geiger. OctNet: Learning deep 3D representations at high resolutions. In *CVPR*, 2017. 2

[39] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11), 2008. 1

[40] R. Salakhutdinov and G. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Artificial Intelligence and Statistics*, 2007. 2

[41] M. Simonovsky and N. Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, 2017. 2

[42] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy. Learnable structured clustering framework for deep metric learning. *CoRR, abs/1612.01213*, 2016. 2

[43] S. Song, H. Lee, and S. Jo. Boundary-enhanced supervoxel segmentation for sparse outdoor lidar data. *Electronics Letters*, 50(25), 2014. 2

[44] D. Stutz, A. Hermans, and B. Leibe. Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166, 2018. 2

[45] L. P. Tchapmi, C. B. Choy, I. Armeni, J. Gwak, and S. Savarese. SEGCloud: Semantic segmentation of 3D point clouds. *International Conference on 3D Vision*, 2017. 2

[46] M. Van den Bergh, X. Boix, G. Roig, and L. Van Gool. SEEDS: superpixels extracted via energy-driven sampling. *International Journal of Computer Vision*, 111(3), 2015. 2

[47] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *CVPR*, 2014. 3, 4

[48] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin. Deep metric learning with angular loss. In *ICCV*. IEEE, 2017. 2

[49] S. Wang, S. Suo, W.-C. M. A. Pokrovsky, and R. Urtasun. Deep parametric continuous convolutional neural networks. In *CVPR*, 2018. 2, 8

[50] X. Xiong, D. Munoz, J. A. Bagnell, and M. Hebert. 3-d scene analysis via sequenced predictions over points and regions. In *ICRA IEEE*. IEEE, 2011. 1

[51] C. Xu and J. J. Corso. Evaluation of super-voxel methods for early video processing. In *CVPR*, 2012. 2

[52] J. Yao, M. Boben, S. Fidler, and R. Urtasun. Real-time coarse-to-fine topologically preserving segmentation. In *CVPR*, 2015. 2

[53] X. Ye, J. Li, H. Huang, L. Du, and X. Zhang. 3D recurrent neural networks with context fusion for point cloud semantic segmentation. In *ECCV*. Springer, 2018. 2

[54] X. Ye, J. Li, H. Huang, L. Du, and X. Zhang. 3d recurrent neural networks with context fusion for point cloud semantic segmentation. In *ECCV*, 2018. 8

[55] T. Zhang et al. Some sharp performance bounds for least squares regression with l1 regularization. *The Annals of Statistics*, 37(5A), 2009. 4

[56] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2017. 2