



HAL
open science

Supervised segmentation with graph-structured deep metric learning

Loic Landrieu, Mohamed Boussaha

► **To cite this version:**

Loic Landrieu, Mohamed Boussaha. Supervised segmentation with graph-structured deep metric learning. ICML Workshop on Learning and Reasoning with Graph-Structured Representations, Jun 2019, Long Beach (CA), United States. hal-03016114

HAL Id: hal-03016114

<https://hal.science/hal-03016114>

Submitted on 20 Nov 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Supervised Segmentation with Graph-Structured Deep Metric Learning

Loic Landrieu^{*1} Mohamed Boussaha^{†1}

Abstract

We present a fully-supervised method for learning to segment data structured by an adjacency graph. We introduce the *graph-structured contrastive loss*, a loss function structured by a ground truth segmentation. It promotes learning vertex embeddings which are homogeneous within desired segments, and have high contrast at their interface. Thus, computing a piecewise-constant approximation of such embeddings produces a graph-partition close to the objective segmentation. This loss is fully backpropagable, which allows us to learn vertex embeddings with deep learning algorithms. We evaluate our methods on a 3D point cloud oversegmentation task, defining a new state-of-the-art by a large margin. These results are based on the published work of Landrieu & Boussaha (2019).

1. Introduction

We consider the problem of learning to segment data points into meaningful groups. More precisely, we consider the case in which such points are linked by a *sparse* graph-structure, and each point is attributed with expressive features. In this case, segmentation can be viewed as a graph partitioning problem based on learned vertex embeddings.

The task of segmentation has been extensively studied for images (Achanta et al., 2012), 3D point clouds (Lin et al., 2018; Papon et al., 2013), and community retrieval (Fortunato, 2010). A common roadblock is that segmentation operators are not backpropagable. The first problem is that the codomain of such operators is the set of vertex partitions for which no simple metric can be used to compute derivatives. Furthermore, graph partitions generally rely on computing connected components, which is highly discontinuous as a single edge can completely overhaul the partition. These limitations prevent directly using segmentation metrics for learning point embeddings.

¹Univ. Paris-Est, IGN-ENSG, LaSTIG, *STRUDEL, †ACTE, Saint-Mandé, France. Correspondence to: Loic Landrieu <loic.landrieu@ign.fr>.

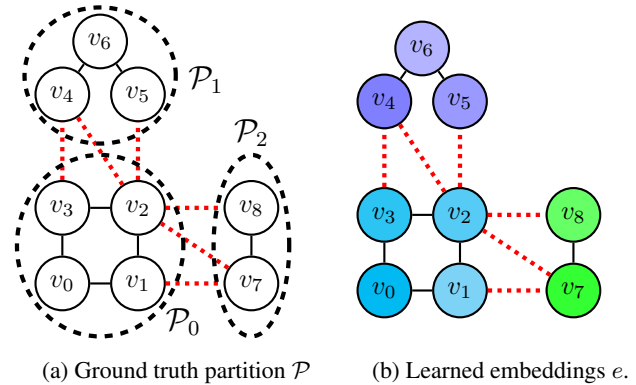


Figure 1. Illustration of our method: vertex embeddings (b) are learned such that they are homogeneous within ground truth segments \mathcal{P} (a), and with high contrast at transition edges (in red).

These issues are addressed by Jampani et al. (2018) for images by using *soft-affectations*, which render the segmentation process continuous and allow for backpropagation. However, these improvements come with strong assumptions on the uniform size and distribution of the segments. While these are reasonable for computing superpixels, our objective is to allow for adaptive segment size and density in order to deal with other types of data.

Liu et al. (2018) proposes a new loss function, called the *segmentation-aware loss* (SEAL), for learning pixel embeddings while taking into account their influence on the quality of the segmentation. In this paper, we propose an improved way of incorporating guidance from segmentation, which is simultaneously more stable and accurate.

Community discovery methods are largely based on graph topology analysis rather than point embeddings, and hence are beyond the scope of this paper. Likewise, the random forest-based supervised graph partitioning algorithm of Reas et al. (2018) aims at learning to recognize transitions between segments rather than learning vertex embeddings. Finally, the work of Liao et al. (2018) uses graph partitioning to help classification rather than learn to segment.

2. Method

We propose a two step approach to graph partitioning supervised by a ground truth segmentation. First, we compute vertex embeddings which are homogeneous within segments and present high contrasts at their border; second, we compute a piecewise-constant approximation of these embeddings with respect to an adjacency graph, as represented in Figure 1. As suggested by Wang et al. (2014), we bound the embeddings to the unit sphere to prevent their collapse during learning.

We consider a set V of data points, whose adjacency structure is encoded by the graph $G = (V, E)$, with $E \subset V \times V$ the set of edges. Throughout this paper, we will assume that this adjacency structure is *sparse*, in the sense that $|E| \ll |V|^2$. Note that this is not necessary for the mathematical derivations of the method. However, the proposed method would not be well-suited for a segmentation problem with a complete adjacency structure.

For a partition $\mathcal{U} = (U_1, \dots, U_K)$ of V , we denote $E_{\text{trans}}(\mathcal{U})$ its set of transition edges, *i.e.* linking different elements of \mathcal{U} : $E_{\text{trans}}(\mathcal{U}) = \{(u, v) \in E \mid u \in U_i, v \in U_j, i \neq j\}$.

Generalized Minimal Partition. We associate to each vertex v an embedding e_v in the m -dimensional unit sphere $\mathbb{S}_m = \{x \in \mathbb{R}^m \mid \|x\| = 1\}$. For such embeddings, we can partition V into the constant connected component of a piecewise-constant approximation of e with respect to graph G . Such approximation can be defined as the solution of the Generalized Minimal Partition Problem (GMPP) introduced by Landrieu & Obozinski (2017):

$$e^* \in \arg \min_{f \in \mathbb{S}_m^V} \sum_{i \in V} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} [f_i \neq f_j], \quad (1)$$

with $w \in \mathbb{R}_+^E$ the edges' weight and $[x \neq y]$ the Iverson's bracket equal to 0 if $x = y$ and 1 otherwise. To encourage splitting 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 $\lambda, \sigma \in \mathbb{R}^+$. This problem is noncontinuous, nondifferentiable, and nonconvex, and hence hard to solve. However, good approximate solutions can be efficiently computed with the $\ell=0$ cut-pursuit algorithm of Landrieu & Obozinski (2017). Thus, a given embedding e defines a GMPP, whose approximate solution is e^* , whose constant connected components defines a segmentation, which we denote $\mathcal{S}^{(e)}$.

Undersegmentation Error. Given a proposed partition $\mathcal{S} = \{S_1, \dots, S_L\}$ of G , one can define its agreement with a ground truth segmentation \mathcal{P} through the *undersegmentation error* \mathcal{L} (Levinshtein et al., 2009), which sums over each segment S_l the number of vertices which are not in the *majority true segment*, *i.e.* the element of \mathcal{P} with the

largest overlap with S_l :

$$\mathcal{L}(P, S) = \frac{1}{|V|} \sum_{l=1}^L \min_{P \in \mathcal{P}} |S_l \setminus P|.$$

Learning Embeddings for Segmentation. Our objective is to learn a vertex embedding function $\xi : V \mapsto \mathbb{S}_m$ such that $\xi(V)$ is homogeneous within the segments of the ground truth segmentation \mathcal{P} , and with high contrast at transition edges $E_{\text{trans}}(\mathcal{P})$. This property encourages $E_{\text{trans}}(\mathcal{S}^{(\xi(V))})$ —the transition edges between constant connected components of the piecewise approximation of e —to be close to $E_{\text{trans}}(\mathcal{P})$. The function ξ is typically a neural network operating on features of the data points corresponding to the vertices of V . This can be for example the color of the pixels of an image, or the local geometry/radiometry of 3D points in a point cloud. Furthermore, these features can be computed from the neighbors of each vertex in G , allowing for the use of a wide range of networks such as convolution-based architectures.

Graph-Structured Contrastive Loss. The naive way to learn such an embedding function ξ would be to minimize the undersegmentation error $\mathcal{L}(\mathcal{S}^{(\xi(V))}, \mathcal{P})$ directly. However, because the optimization problem defined in (1) is non-continuous and nonconvex, it is difficult, if not impossible, to backpropagate through its minimization. As discussed earlier, the constant connected component operator is not backpropagable either. Furthermore, the undersegmentation error would favor very granular partitions as it doesn't penalize high segment counts.

Consequently, we introduce ℓ the graph-structured contrastive loss, a surrogate loss function to the undersegmentation error, which operates on edges instead vertices and allows for backpropagation:

$$\ell(e, \mathcal{P}) \cdot |E| = \sum_{(u,v) \in E \setminus E_{\text{trans}}(\mathcal{P})} \phi(e_u - e_v) + \sum_{(u,v) \in E_{\text{trans}}(\mathcal{P})} \mu_{u,v}^{(e)} \psi(e_u - e_v),$$

with ϕ (resp. ψ) a function favoring similarity (resp. contrast), and $\mu_{i,j}^{(e)} \in \mathbb{R}^{E_{\text{trans}}}$ a weight on transition edges, discussed later. A vertex embedding function minimizing this loss will be uniform within elements of \mathcal{P} and have high contrasts at $E_{\text{trans}}(\mathcal{P})$. Consequently, $E_{\text{trans}}(\mathcal{S}^{(e)})$ should be close to $E_{\text{trans}}(\mathcal{P})$. Our proposed loss is related to the contrastive loss of Chopra et al. (2005) and the triplet loss popularized by both Hoffer & Ailon (2015) and Wang et al. (2014). However, our method takes advantage of the adjacency structure. This allows us to bypass the problem of *example picking* altogether. Indeed, the positive and negative examples are directly determined by the graph structure, instead of computationally intensive hard example mining.

We chose ϕ —the function promoting intra-segment homogeneity—as $\phi(x) = \delta(\sqrt{\|x\|^2/\delta^2 + 1} - 1)$ (represented in Figure 2). This means that the first term of ℓ is the

(pseudo)-Huber graph-total variation on the non-transition edges (Huber et al., 1973; Charbonnier et al., 1997), promoting smooth homogeneity of embeddings within segments.

With $\psi(x) = \max(1 - \|x\|, 0)$, the second part of ℓ is the opposite of the truncated graph-total variation (Zhang et al., 2009) on the transition edges. It penalizes similar embeddings at the edges between ground truth segments. As the embeddings are spherically-bound, we threshold this function for differences larger than 1 (corresponding to a 60 degree angle). In other words, ψ encourages vertices linked by a transition edge to have embeddings with an euclidean distance of 1, but does not push for a larger difference.

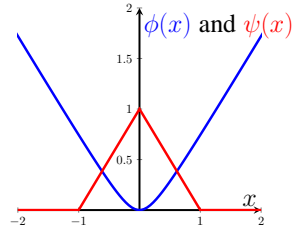


Figure 2. The functions ϕ (in blue) and ψ (in red) used in the graph-structured contrastive loss.

Note that a loss of 0 can be achieved by any embeddings which are exactly constant within ground truth segments and have a difference of at least 1 between adjacent segments. The four-color theorem (Gonthier, 2008) tells us that this is always possible as long as the dimension of the embeddings is at least 3. However, because embeddings are computed by ξ , which operates on vertex features, there needs to be a recognizable pattern at the border between segments in order for our method to detect a transition.

Cross-partition Weighting. Note that without an appropriate edge weighting scheme, this loss will only encourage high accuracy in recovering transition edges. However, the influence of each edge can be vastly different in terms of undersegmentation error \mathcal{L} . Indeed, a single missed edge can result in the erroneous fusion of large adjacent segments. In order for ℓ to better represent \mathcal{L} , we choose the edge weights $\mu^{(e)} \in \mathbb{R}_+^{E_{\text{intra}}(\mathcal{P})}$ to reflect this influence.

To this end, we introduce the *cross-partition graph* $\mathcal{G} = (\mathcal{C}, \mathcal{E})$, represented in Figure 3 and defined as the adjacency graph of the cross-partition between \mathcal{P} and $\mathcal{S}^{(e)}$ considering only transition edges $E_{\text{trans}}(\mathcal{P})$:

$$\begin{aligned} \mathcal{C} &= \{P \cap S \mid P \in \mathcal{P}, S \in \mathcal{S}^{(e)} \text{ and } P \cap S \neq \emptyset\} \\ \mathcal{E} &= \{(U, V) \in \mathcal{C}^2 \mid U \times V \cap E_{\text{trans}} \neq \emptyset\}. \end{aligned}$$

We associate the following weight $M_{U,V}$ to each edge (U, V) of \mathcal{E} and $\mu_{u,v}$ to each transition edge:

$$\begin{aligned} M_{U,V}^{(e)} &= M_0 \min(|U|, |V|) \text{ for } (U, V) \in \mathcal{E} \\ \mu_{u,v}^{(e)} &= \frac{M_{U,V}^{(e)}}{|U \times V \cap E_{\text{trans}}|} \text{ for } (u, v) \in U \times V \cap E_{\text{trans}}. \end{aligned}$$

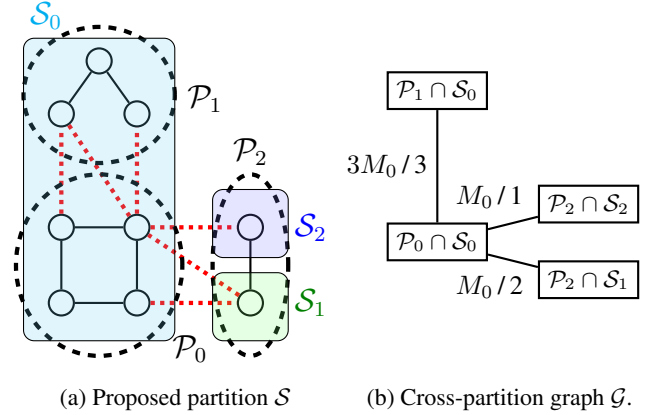


Figure 3. Illustration of the proposed superedge weighting scheme. In (a), we represent an erroneous proposed partition \mathcal{S} with 3 segment. In (b) we represent the cross partition graph \mathcal{G} with the edge weights $M_{U,V} / |U \times V \cap E_{\text{trans}}|$.

with μ a parameter of the model. By definition of \mathcal{E} , two segments U and V of \mathcal{C} linked by a superedge $(U, V) \in \mathcal{E}$ are in two different ground truth segments. The undersegmentation error caused by the erroneous fusion of these two components is proportional to $\min(|U|, |V|)$. This error, spread evenly over the edges constituting the transition superedge, determines the edge weights.

This weighting scheme differs from the SEAL strategy (Liu et al., 2018). Indeed, in the latter, the edge weights are shared by all transition edges of a given segment. This favors long interfaces in the loss too strongly, and inadequately handles large segments with multiple interfaces. Finally, SEAL sets the weights to 1 as soon as a border is retrieved, which makes the loss rather unstable and hard to optimize.

Setting $M_0 = |E| / |V|$ gives the same importance to the classification of transition and non-transition edges. Indeed, assuming that most edges are non-transition, we have the sum of non-transition edge weights close to $|E|$, while the sum of transition edge weights is in the order of magnitude of $|V|$. In an oversegmentation setting, M_0 must be set higher to prioritize recovering object borders.

3. Numerical Experiments

We present numerical illustrations of our approach for 3D point cloud oversegmentation. To this end we consider two different datasets: S3DIS, composed of dense indoor scans (Armeni et al., 2016), and vKITTI3D, a virtual dataset of sparse outdoor point clouds (Engelmann et al., 2017; Gaidon et al., 2016). The first one has both object and semantic label annotations. For the second one, we define the ground truth partition \mathcal{P} as the connected components of the semantic labels. Both datasets are composed of 6 independent parts, which allows us to perform 6-fold cross-validation.

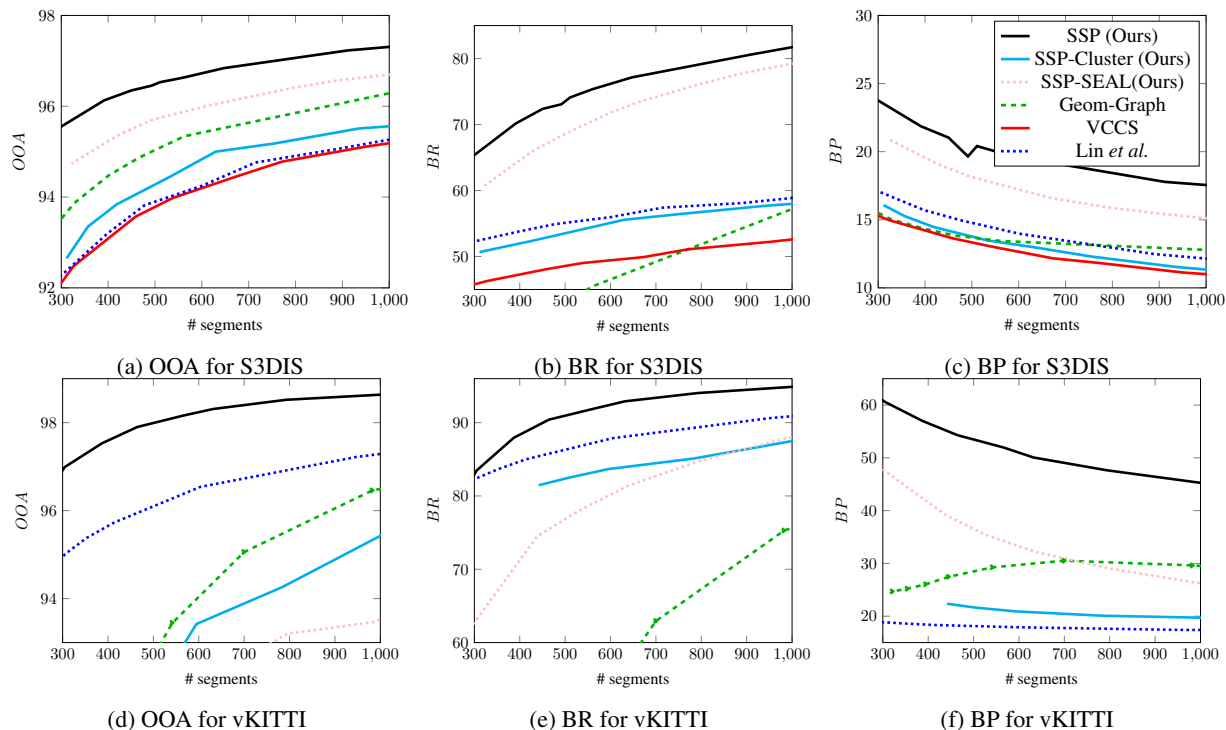


Figure 4. Performance of the different algorithms on the 6-fold S3DIS dataset (a, b, c), and the 6-fold vKITTI3D dataset (d, e, f).

Note that for considerations of simplicity and efficiency, we relax the optimization domain of (1) from \mathbb{S}_m to \mathbb{R}^m , while still learning spherical embeddings. While this can lead to suboptimal partitions, the segmentation retrieved are still relevant. We set the dimensions of the embeddings to 4, and M_0 to $5|E|/|V|$, hence favoring oversegmentations. The embedding function is a small PointNet-like network (Qi et al., 2017) operating on the 20 nearest neighbors of each point. More details in the appendix.

In Figure 4, we report the performance of our algorithm according to three segmentation metrics: OOA, BR, and BP. OOA denotes the Oracle Overall Accuracy, *i.e.* the OA of the oracle classification algorithm associating the majority label to each segment of the proposed partition \mathcal{S} . Note that the OOA is a higher bound on the pointwise overall accuracy of any classification algorithm operating on the segments. The OOA is also closely linked to the undersegmentation error, but adds a semantic component. BP (resp. BR) denotes the precision (resp. recall) of the predicted transition edges $E_{\text{intra}}(\mathcal{S})$ compared to $E_{\text{intra}}(\mathcal{P})$ with a tolerance of one edge. We denote our method by **SSP** for *Supervised SuperPoints* and compare our approach to the following methods:

SSP-cluster is our adaptation of the soft partition approach of Jampani et al. (2018) to the 3D setting.

SSP-SEAL uses the same framework as **SSP**, but with the cross-partition weights replaced by the SEAL weighting strategy (Liu et al., 2018). Note that this is *not* equivalent

to the framework of Liu et al. (2018), as they use a different loss and clustering algorithm.

Geom-graph is the graph-based method introduced by Guinard & Landrieu (2017) solving (1) on handcrafted features (Demantke et al., 2011) instead of learned ones.

VCCS is the octree-structured cluster-based method introduced by Papon et al. (2013).

Lin et al. is the adaptive resolution graph-based method introduced by Lin et al. (2018).

We observe that for the large S3DIS dataset (600 Mpoints), supervised methods provide considerably better results. In particular, our method **SSP** obtains better accuracy with 300 segments than the state-of-the-art method of Lin et al. with 1500 segments. The advantages for border recall and precision are even more significant. For the smaller vKITTI3D dataset (15 Mpoints), Lin et al. obtain better results than all supervised methods except our approach. Illustration of the results as well as more details on the models and metrics are given in the appendix.

Conclusion

We presented a framework for learning to segment graph-structured data with neural networks. Our new loss is fully backpropagable and indirectly takes the undersegmentation error into account. We assess its efficiency on two large-scale point cloud oversegmentation benchmarks. We demonstrate a significant improvement over unsupervised

methods of the state-of-the-art and our own implementations of other supervised methods. All codes will be released at the following URL: github.com/loiccland/superpoint_graph. Future works includes applying our method to other graph-structured data types such as images or relationship graphs and solving the GMPP with the spherical domain constraint.

References

- Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S., et al. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 34(11), 2012.
- Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I. K., Fischer, M., and Savarese, S. 3d semantic parsing of large-scale indoor spaces. In *CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016. doi: 10.1109/CVPR.2016.170. URL <https://doi.org/10.1109/CVPR.2016.170>.
- Charbonnier, P., Blanc-Féraud, L., Aubert, G., and Barlaud, M. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on Image Processing*, 6(2), 1997.
- Chopra, S., Hadsell, R., and LeCun, Y. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, volume 1. IEEE, 2005.
- Delaunay, B. et al. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800): 1–2, 1934.
- Demantke, J., Mallet, C., David, N., and Vallet, B. Dimensionality based scale selection in 3d lidar point clouds. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci*, 38(5):W12, 2011.
- Engelmann, F., Kontogianni, T., Hermans, A., and Leibe, B. Exploring spatial context for 3d semantic segmentation of point clouds. In *ICCV Workshops*, 2017. doi: 10.1109/ICCVW.2017.90. URL <https://doi.org/10.1109/ICCVW.2017.90>.
- Engelmann, F., Kontogianni, T., Schult, J., and Leibe, B. Know what your neighbors do: 3d semantic segmentation of point clouds. In *GMDL Workshop, ECCV*, 2018.
- Fortunato, S. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.
- Gaidon, A., Wang, Q., Cabon, Y., and Vig, E. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016.
- Gonthier, G. Formal proof—the four-color theorem. *Notices of the AMS*, 55(11), 2008.
- Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT press Cambridge, 2016.
- Guinard, S. and Landrieu, L. Weakly supervised segmentation-aided classification of urban scenes from 3d lidar point clouds. In *ISPRS Workshop*, 2017.
- Hoffer, E. and Ailon, N. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015.
- Huber, P. J. et al. Robust regression: asymptotics, conjectures and monte carlo. *The Annals of Statistics*, 1(5), 1973.
- Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.
- Jaderberg, M., Simonyan, K., Zisserman, A., et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, 2015.
- Jampani, V., Sun, D., Liu, M., Yang, M., and Kautz, J. Superpixel sampling networks. In *ECCV*, 2018. doi: 10.1007/978-3-030-01234-2_22. URL https://doi.org/10.1007/978-3-030-01234-2_22.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *ICLR*, 2015.
- Landrieu, L. and Boussaha, M. Point cloud oversegmentation with graph-structured deep metric learning. In *CVPR*, 2019.
- Landrieu, L. and Obozinski, G. Cut pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs. *SIAM Journal on Imaging Sciences*, 10(4), 2017.
- Levinshtein, A., Stere, A., Kutulakos, K. N., Fleet, D. J., Dickinson, S. J., and Siddiqi, K. Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12), 2009.
- Liao, R., Brockschmidt, M., Tarlow, D., Gaunt, A. L., Urtasun, R., and Zemel, R. Graph partition neural networks for semi-supervised classification. *International Conference on Learning Representations Workshop (ICLR)*, 2018.
- Lin, Y., Wang, C., Zhai, D., Li, W., and Li, J. Toward better boundary preserved supervoxel segmentation for 3d point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 143, 2018. ISSN 0924-2716. doi: <https://doi.org/10.1016/j.isprsjprs.2018.05.004>. URL <http://www.sciencedirect.com/science/article/pii/S0924271618301370>.
- Liu, M.-Y., Tuzel, O., Ramalingam, S., and Chellappa, R. Entropy rate supervoxel segmentation. In *CVPR*. IEEE, 2011.
- Liu, W.-C. T. M.-Y., Shao-Yi, V. J. D. S., Yang, C. M.-H., and Kautz, J. Learning superpixels with segmentation-aware affinity loss. In *CVPR*. IEEE, 2018.
- Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.
- Papon, J., Abramov, A., Schoeler, M., and Wörgötter, F. Voxel cloud connectivity segmentation - supervoxels for point clouds. In *CVPR*, 2013. doi: 10.1109/CVPR.2013.264. URL <https://doi.org/10.1109/CVPR.2013.264>.
- Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CVPR, IEEE*, 1(2), 2017.
- Reas, R., Ash, S., Barton, R., and Borthwick, A. Superpart: Supervised graph partitioning for record linkage. In *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 387–396. IEEE, 2018.
- Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., and Wu, Y. Learning fine-grained image similarity with deep ranking. In *CVPR*, 2014.

Wu, Y. and He, K. Group normalization. *ECCV*, 2018.

Zhang, T. et al. Some sharp performance bounds for least squares

regression with l1 regularization. *The Annals of Statistics*, 37 (5A), 2009.

SUPPLEMENTARY MATERIALS

A. 3D Point Embedding

In this section, we describe the embedding function ξ used in the 3D point cloud oversegmentation application. This function associate to each point a spherical m -dimensional embedding e_i characterizing its point-features (position, color, etc.) and the geometry and radiometry of its local neighborhood. To this end, we introduce the Local Point Embedder (LPE), a lightweight network inspired by PointNet (Qi et al., 2017). 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 . It normalizes the neighbors' coordinates around p_i , and such that the standard deviation of the point's position is equal to 1 (4). Then, this neighborhood is rotated around the z axis with a 2×2 rotation matrix computed by small PointNet network PTN (5). As advocated by (Jaderberg et al., 2015), 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: $[\Omega_{x,x}, \Omega_{x,y}, \Omega_{y,x}, \Omega_{y,y}]$) (6). 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.

$$\text{rad} = \text{std}(P_i) \quad (2)$$

$$\Omega = \text{PTN}(\tilde{P}_i) \quad (3)$$

$$P'_i = (P_i - p_i)/\text{rad} \quad (4)$$

$$\tilde{P}_i = \{p \times \Omega \mid p \in P'_i\} \quad (5)$$

$$\tilde{p}_i = [p_i^{(z)}, \text{rad}, \Omega] \quad (6)$$

Local Point Embedder: The LPE network computes a normalized embedding from two inputs: a point-feature x_i and a set-feature X_i . As in PointNet (Qi et al., 2017), 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 (Nair & Hinton, 2010)), normalization (batch (Ioffe & Szegedy, 2015)), and so on. The resulting set-features are then max-pooled into a point-feature, which is concatenated with the input point-feature. The resulting vector is processed through another multi-layer perceptron MLP_2 (8), and finally normalized on the unit sphere.

The embeddings e_i are computed for each point i of C

through a shared LPE (9). 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 .

$$L_2(\cdot) = \cdot / \|\cdot\| \quad (7)$$

$$\text{LPE}(X_i, x_i) = L_2(\text{MLP}_2([\max(\text{MLP}_1(X_i)), x_i])) \quad (8)$$

$$e_i = \text{LPE}([\tilde{P}_i, R_i], [\tilde{p}_i, r_i]) \quad (9)$$

A.1. Implementation Details

We use a modified version of the ℓ_0 -cut pursuit algorithm¹(Landrieu & Obozinski, 2017), with two main differences:

- to prevent the creation of many small segments in regions of high contrast, we merge components greedily with respect to the MGPP energy (1), as long as they are smaller than a given threshold;
- we heuristically improved the forward step (1) from (Landrieu & Obozinski, 2017), such that the regularization strength increases geometrically by a factor (of 0.7) along the iterations. This helps improve the quality of the lower optima retrieved, and consequently the graph partition.

To limit the spatial extent of the segment we concatenate to the points' embeddings their 3D coordinates in (1) multiplied by a parameter α_{spatial} , in the manner of (Achanta et al., 2012). 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.

B. Oversegmentation Metrics

There are many standard metrics which assess the quality of point cloud oversegmentation. 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. In the literature, these measures are defined with respect to *boundary pixels* (Papon et al., 2013) or points (Lin et al., 2018). However, we argue that transition occurs *between* points and not *at* points for point clouds. Consequently, we define $E_{\text{trans}}^{\text{pred}}$ 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

¹<https://github.com/loicland/cut-pursuit>

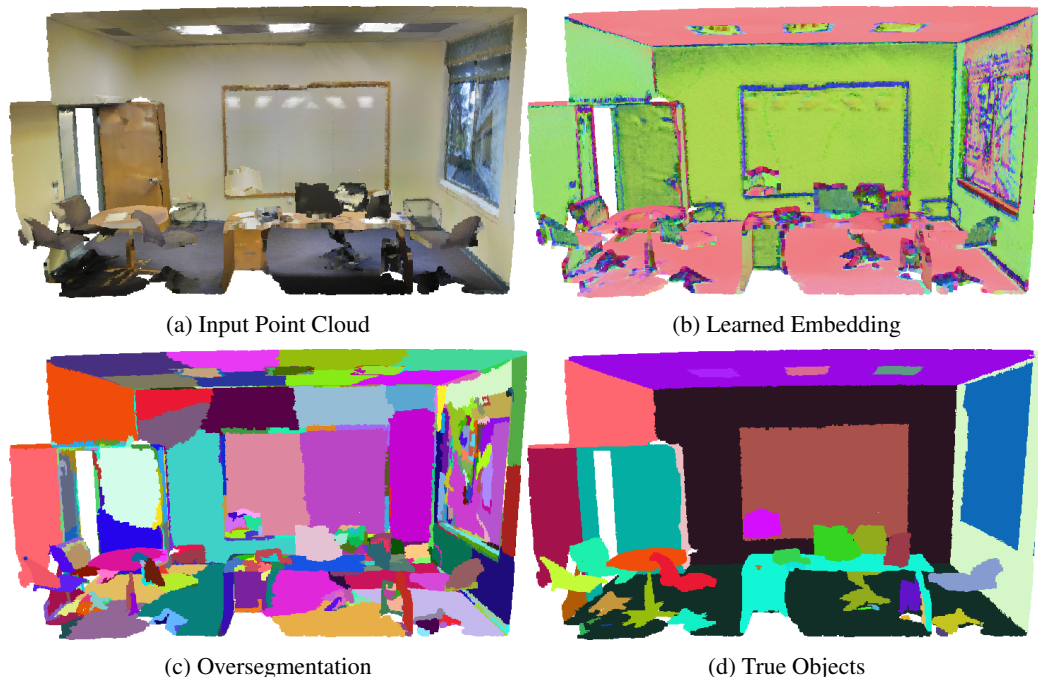


Figure 5. Illustration of our framework on a hard-to-segment scene with a white board on a white wall: a colored point cloud is given as input (a), an embedding is computed for each point (b), which allows a clustering technique to compute an oversegmentation (c), which closely follows the ground truth (d). The embeddings are projected into a 3-dimensional space to allow color visualization.

border for the latter to be considered retrieved. We set this distance to 1 edge, which leads us to define $E_{\text{trans}}^{(1)}$ the set of inter-edges expanded to all directly adjacent edges in E :

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

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

$$BR = \frac{|E_{\text{trans}}^{\text{pred}} \cap E_{\text{trans}}^{(1)}|}{|E_{\text{trans}}^{\text{pred}}|}, \quad BP = \frac{|E_{\text{trans}}^{\text{pred}} \cap E_{\text{trans}}^{(1)}|}{|E_{\text{trans}}^{(1)}|}.$$

To assess object purity we define the *Oracle Overall Accuracy* (OOA). 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:

$$l^{\text{oracle}}(S) = \text{mode} \{l_i \mid i \in S\}$$

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

with $[x = y]$ the function equal to 1 if $x = y$ and 0 otherwise. Note that the OOA is closely related to the ASA (Liu et al., 2011), but consider the majority labels of all points within a superpixel rather than the label of the objects with most overlap. This metric is also more fair than

the undersegmentation error (Levinshtein et al., 2009) for other methods such as (Guinard & Landrieu, 2017), or our cluster-based approach, as they do not try to retrieve objects directly, but rather regions of C with homogeneous semantic labeling.

In Figure 6, we show the oversegmentation results of our method and the competing algorithms on vKITTI3D and S3DIS datasets. We observe that our supervised partition framework produces superpoints of adaptive sizes which closely follow hard-to-segment objects such as white boards or sidewalks. We also notice that the embeddings learn to ignore certain form of intra-object variability of geometry and radiometry. In particular, the lamp reflections on the white boards are almost completely ignored by the embeddings. Even more interestingly, the embeddings of trees are homogeneous despite the significant variability between leafs and trunks. As a consequence, the trees are segmented into one component while the other methods produces many dubious superpoints.

C. Ablation Study

We present an ablation study to empirically justify some of our design choices. 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

parameter	shorthand	section	S3DIS	vKITTI
Local neighborhood size	k	3.1		20
# parameters	-	-		13,816
LPE configuration	-	3.1	[32,128],[64,32,32,m]	
ST configuration	-	3.1	[16,64],[32,16,4]	
Embeddings dimension	m	3.1		4
Adjacency graph	G	3.2	5-nn	5-nn + Delaunay
exponential edge factor	σ	3.2.1		0.5
intra-edge factor	$\tilde{\mu}$	3.2.3		5
spatial influence	α_{spatial}	3.4	0.2	0.02
smallest superpoint	$n_{\text{min}}^{(1)}$	3.4	40	10
epochs	-	-		50
decay event	-	-		20,35,45

Table 1. Configuration of the embedding network for the S3DIS and vKITTI datasets.

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 replaced 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 (Engelmann et al., 2018) (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.

D. Models configuration

Our supervised oversegmentation model has a number of critical hyper-parameters to tune, given in Table 1. We detail here the rationale behind our choices.

Local neighborhood and adjacency graphs: For both datasets, we find that setting the local neighborhood size to 20 was enough for embeddings to successfully detect objects’ borders. Combined with our lightweight structure, this results in a very low memory load overall. The adjacency graph G requires more attention depending on the dataset. For the dense scans of S3DIS, the 5-nearest neighbors adjacency structure was enough to capture the connectivity of the input clouds. For the sparse scans of vKITTI, we added Delaunay edges (Delaunay et al., 1934) (pruned at 50 cm) such that parallel scans lines would be connected.

Networks configuration: For the LPE and the PointNet structure in the spatial transform, we find that shallow and wide architectures works better than deeper networks. We give in Table 1 the size of the linear layers, before and after the maxpool operation. Over 250,000 points can be

embedded simultaneously on 11GB RAM in the training step, while keeping track of gradients.

Intra-edge factor: The graph-structured contrastive loss presented in Section 2 requires setting a weight μ determining the influence of inter-edges with respect to intra-edge. Since most edges of G are intra-edges in practice, we define $\tilde{\mu}$ such that $\mu = \tilde{\mu}c$ with $c = |E|/|V|$ the average connectivity of G . Note that c can be determined directly from the construction of the adjacency graph (it is equal to k in a k -nearest neighbor graph for example). A value of $\tilde{\mu} = 1$ means that the total influence in ℓ of inter-edges and intra-edges are identical. Since we are interested in oversegmentation, we set $\tilde{\mu}$ to 5 in all our experiments, but note that the network is not very sensitive to this parameter, as demonstrated experimentally: a value of $\tilde{\mu} = 3$ gives a relative performance of $(-0.2, -0.6, +1.5)$ while a value of 8 gives $(+0.1, -0.5, +1.4)$.

Regularization Strength: The generalized minimal partition problem defined in Section 2 requires setting the regularization strength factor λ , determining the cost of edges crossing superpoints. We remark that the LPE produces embeddings of points with an euclidean distance of at least 1 over predicted objects’ borders. Some calculus shows us that for a $\lambda \leq 1/(2c)$, the solution e^* of (1) should predict superpoints borders at all edges whose vertices have a difference of embeddings of at least 1 (note that there is no guarantee that the greedy ℓ_0 -cut pursuit algorithm will indeed predict a border). We use this value to define a normalized regularization strength $\tilde{\lambda}$ such that $\lambda = \tilde{\lambda}/(4c)$, whose default value is 1.

Regularization path: To obtain the regularization paths in Figure 4, we first train the network with a regularization strength of $\tilde{\lambda} = 1$ (see Section 2). We then compute partitions with $\tilde{\lambda}$ varying from 0.2 to 6 with no fine-tuning required.

Smallest superpoint: To automatically select a minimal superpoint size (in number of points) appropriate to the coarseness of the segmentation, we heuristically set:

$$n_{\min}^{\tilde{\lambda}} = \left\lceil \max \left(\frac{1}{2} n_{\min}^{(1)}, n_{\min}^{(1)} + \frac{1}{2} n_{\min}^{(1)} \log(\tilde{\lambda}) \right) \right\rceil$$

where $n_{\min}^{(1)}$ is a dataset-specific minimum superpoints size for $\tilde{\lambda} = 1$. For example, for $n_{\min}^{(1)} = 50$, the smallest superpoint allowed for a small regularization strength $\tilde{\lambda} = 0.2$ will be 33, while it is 70 for the coarse partition obtained with $\tilde{\lambda} = 6$. While specific applications may require setting up this variable manually, this allowed us to produce the regularization paths in Figure 4 while only varying $\tilde{\lambda}$.

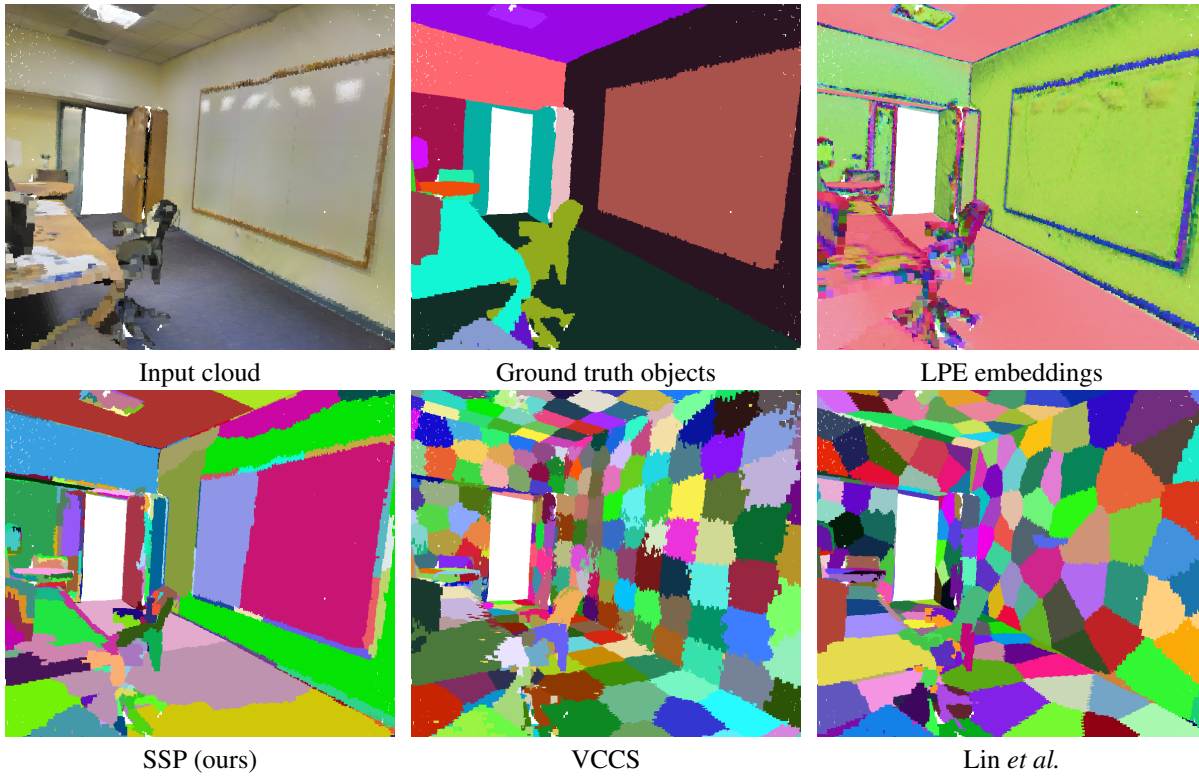
Optimization: Given the small size of our network, we train it for a short number of epochs (see Table 1), with decay events set at 0.7. We use Adam optimizer (Kingma & Ba, 2015) with gradient clipping at 1 (Goodfellow et al., 2016). Training takes around 2 hours per fold on our 11GB

VRAM 1080Ti GPU.

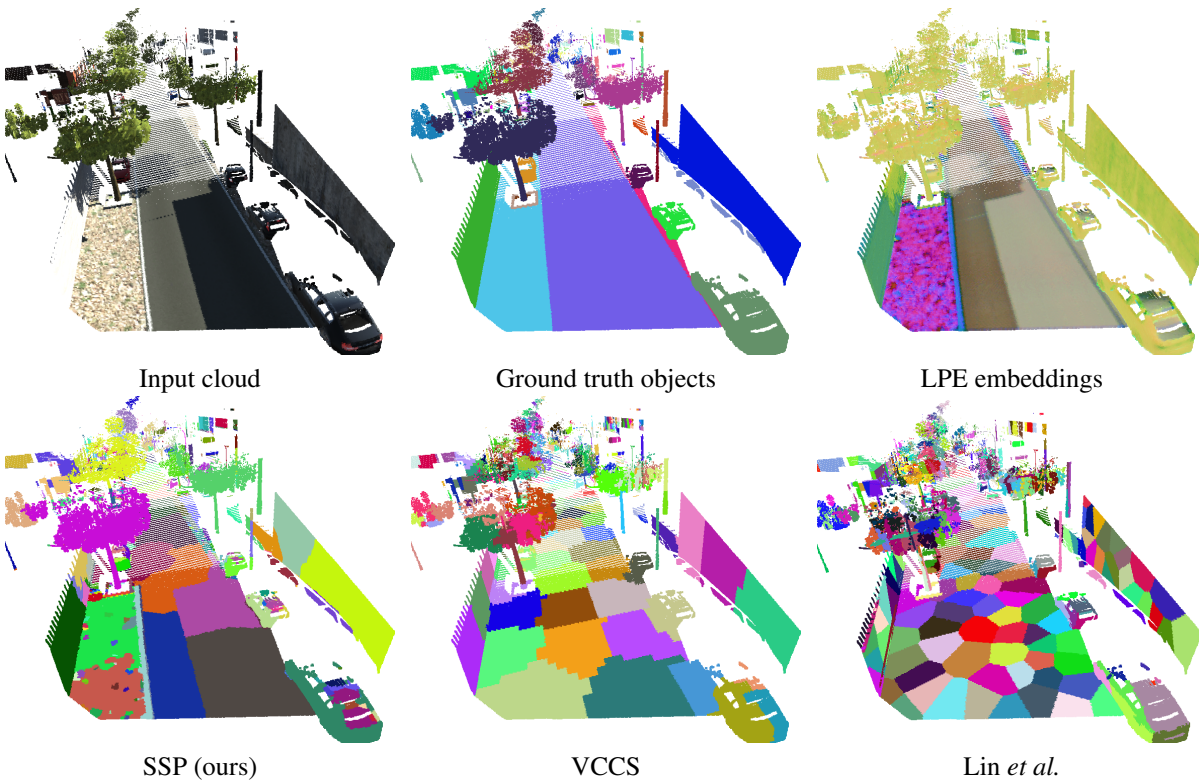
Mini-batches: For graph-based clustering, the training phase processes batches of 16 point clouds at once, for which a subgraph of size 10 000 points is extracted. For the clustering-based segmentation, which is more memory intensive, and since subgraphs have to be larger to be meaningfully covered by the initial voxels, we set a batch size of 1 and a subgraph of 100 000. As a consequence, we replace the batchnorm layers of the LPEs by group norms with 4 groups (Wu & He, 2018).

Augmentation: In order to build more robust networks, we added Gaussian noise of deviation 0.03 clamped at 0.1 on the normalized position and color of neighborhood clouds. We also added random rotation of the input clouds for the network to learn rotation invariance. To preserve orientation information, the clouds are rotated as a whole instead of each neighborhood. This allows the spatial transform to detect change in orientation, which can be used to detect borders.

Supervised Segmentation



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



(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.

Supervised Graph-Structured Segmentation with Deep Metric Learning

SUPPLEMENTARY MATERIALS

Anonymous Authors¹

1. 3D Point Embedding

In this section, we describe the embedding function ξ used in the 3D point cloud oversegmentation application. This function associate to each point a spherical m -dimensional embedding e_i characterizing its point-features (position, color, etc.) and the geometry and radiometry of its local neighborhood. To this end, we introduce the Local Point Embedder (LPE), a lightweight network inspired by PointNet (?). 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 . 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 (?), 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.

$$\text{rad} = \text{std}(P_i) \quad (1)$$

$$\Omega = \text{PTN}(\tilde{P}_i) \quad (2)$$

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

$$\tilde{P}_i = \{p \times \Omega \mid p \in P'_i\} \quad (4)$$

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

Local Point Embedder: The LPE network computes a normalized embedding from two inputs: a point-feature x_i and a set-feature X_i . As in PointNet (?), 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 (?)), normalization (batch (?)), 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 .

$$L_2(\cdot) = \cdot / \|\cdot\| \quad (6)$$

$$\text{LPE}(X_i, x_i) = L_2(\text{MLP}_2([\max(\text{MLP}_1(X_i)), x_i])) \quad (7)$$

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

1.1. Implementation Details

We use a modified version of the ℓ_0 -cut pursuit algorithm¹(?), with two main differences:

- to prevent the creation of many small segments in regions of high contrast, we merge components greedily with respect to the MGPP energy (1), as long as they are smaller than a given threshold;
- we heuristically improved the forward step (1) from (?), such that the regularization strength increases geo-

¹<https://github.com/loicland/cut-pursuit>

¹Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review at the ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data. Do not distribute.

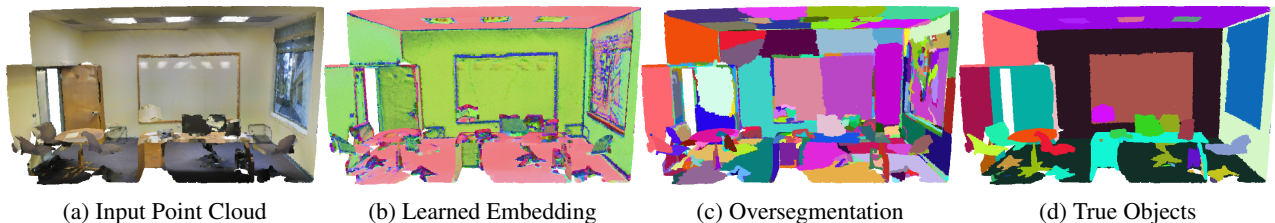


Figure 1. Illustration of our framework on a hard-to-segment scene with a white board on a white wall: a colored point cloud is given as input (a), an embedding is computed for each point (b), which allows a clustering technique to compute an oversegmentation (c), which closely follows the ground truth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

metrically by a factor (of 0.7) along the iterations. This helps improve the quality of the lower optima retrieved, and consequently the graph partition.

To limit the spatial extent of the segment we concatenate to the points’ embeddings their 3D coordinates in (1) multiplied by a parameter α_{spatial} , in the manner of (?). 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.

2. Oversegmentation Metrics

There are many standard metrics which assess the quality of point cloud oversegmentation. 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. In the literature, these measures are defined with respect to *boundary pixels* (?) or points (?). However, we argue that transition occurs *between* points and not *at* points for point clouds. Consequently, we define $E_{\text{trans}}^{\text{pred}}$ 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 $E_{\text{trans}}^{(1)}$ the set of inter-edges expanded to all directly adjacent edges in E :

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

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

$$BR = \frac{|E_{\text{trans}}^{\text{pred}} \cap E_{\text{trans}}^{(1)}|}{|E_{\text{trans}}^{\text{pred}}|}, \quad BP = \frac{|E_{\text{trans}}^{\text{pred}} \cap E_{\text{trans}}^{(1)}|}{|E_{\text{trans}}^{\text{pred}}|}.$$

To assess object purity we define the *Oracle Overall Accuracy* (OOA). 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:

$$l^{\text{oracle}}(S) = \text{mode} \{l_i \mid i \in S\}$$

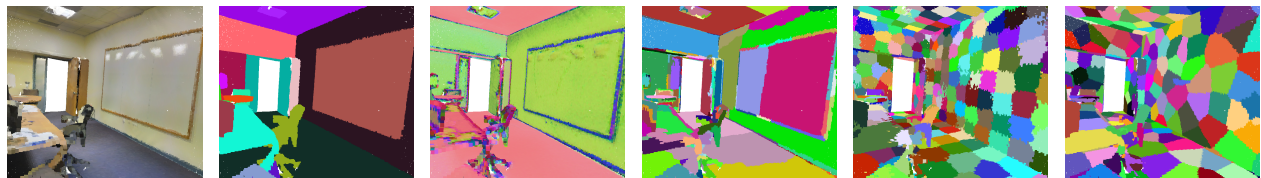
$$OOA = \frac{1}{|C|} \sum_{S \in \mathcal{S}} \sum_{i \in S} [l_i = l^{\text{oracle}}(S)],$$

with $[x = y]$ the function equal to 1 if $x = y$ and 0 otherwise. Note that the OOA is closely related to the ASA (?), but consider the majority labels of all points within a superpixel rather than the label of the objects with most overlap. This metric is also more fair than the undersegmentation error (?) for other methods such as (?), or our cluster-based approach, as they do not try to retrieve objects directly, but rather regions of C with homogeneous semantic labeling. In Figure 2, we show the oversegmentation results of our method and the competing algorithms on vKITTI and S3DIS datasets.

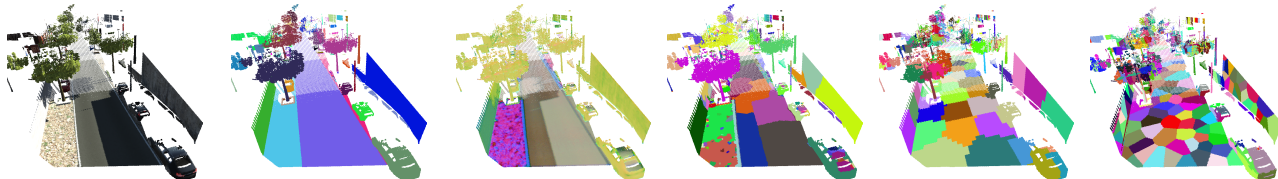
3. Ablation Study

We present an ablation study to empirically justify some of our design choices. 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 replaced 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 (?) (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.



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

Input cloud Ground truth objects LPE embeddings SSP (ours) VCCS Lin *et al.*

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

Figure 2. Illustration of the oversegmentations of our framework, and from competing algorithms.

parameter	shorthand	section	S3DIS	vKITTI
Local neighborhood size	k	3.1		20
# parameters	-	-		13,816
LPE configuration	-	3.1	[32,128],[64,32,32,m]	
ST configuration	-	3.1	[16,64],[32,16,4]	
Embeddings dimension	m	3.1		4
Adjacency graph	G	3.2	5-nn	5-nn + Delaunay
exponential edge factor	σ	3.2.1		0.5
intra-edge factor	$\tilde{\mu}$	3.2.3		5
spatial influence	α^{spatial}	3.4	0.2	0.02
smallest superpoint	$n_{\min}^{(1)}$	3.4	40	10
epochs	-	-		50
decay event	-	-		20,35,45

Table 1. Configuration of the embedding network for the S3DIS and vKITTI datasets.

4. Models configuration

Our supervised oversegmentation model has a number of critical hyper-parameters to tune, given in Table 1. We detail here the rationale behind our choices.

Local neighborhood and adjacency graphs: For both datasets, we find that setting the local neighborhood size to 20 was enough for embeddings to successfully detect objects’ borders. Combined with our lightweight structure, this results in a very low memory load overall. The adjacency graph G requires more attention depending on the dataset. For the dense scans of S3DIS, the 5-nearest neighbors adjacency structure was enough to capture the connectivity of the input clouds. For the sparse scans of vKITTI, we added Delaunay edges (?) (pruned at 50 cm) such that parallel scans lines would be connected.

Networks configuration: For the LPE and the PointNet structure in the spatial transform, we find that shallow and

wide architectures works better than deeper networks. We give in Table 1 the size of the linear layers, before and after the maxpool operation. Over 250,000 points can be embedded simultaneously on 11GB RAM in the training step, while keeping track of gradients.

Intra-edge factor: The graph-structured contrastive loss presented in Section 2 requires setting a weight μ determining the influence of inter-edges with respect to intra-edge. Since most edges of G are intra-edges in practice, we define $\tilde{\mu}$ such that $\mu = \tilde{\mu}c$ with $c = |E|/|V|$ the average connectivity of G . Note that c can be determined directly from the construction of the adjacency graph (it is equal to k in a k -nearest neighbor graph for example). A value of $\tilde{\mu} = 1$ means that the total influence in ℓ of inter-edges and intra-edges are identical. Since we are interested in oversegmentation, we set $\tilde{\mu}$ to 5 in all our experiments, but note that the network is not very sensitive to this parameter, as demonstrated experimentally: a value of $\tilde{\mu} = 3$ gives a

relative performance of $(-0.2, -0.6, +1.5)$ while a value of 8 gives $(+0.1, -0.5, +1.4)$.

Regularization Strength: The generalized minimal partition problem defined in Section 2 requires setting the regularization strength factor λ , determining the cost of edges crossing superpoints. We remark that the LPE produces embeddings of points with an euclidean distance of at least 1 over predicted objects' borders. Some calculus shows us that for a $\lambda \leq 1/(2c)$, the solution e^* of (1) should predict superpoints borders at all edges whose vertices have a difference of embeddings of at least 1 (note that there is no guarantee that the greedy ℓ_0 -cut pursuit algorithm will indeed predict a border). We use this value to define a normalized regularization strength $\tilde{\lambda}$ such that $\lambda = \tilde{\lambda}/(4c)$, whose default value is 1.

Regularization path: To obtain the regularization paths in Figure 4, we first train the network with a regularization strength of $\tilde{\lambda} = 1$ (see Section 2). We then compute partitions with $\tilde{\lambda}$ varying from 0.2 to 6 with no fine-tuning required.

Smallest superpoint: To automatically select a minimal superpoint size (in number of points) appropriate to the coarseness of the segmentation, we heuristically set:

$$n_{\min}^{\tilde{\lambda}} = \left[\max \left(\frac{1}{2} n_{\min}^{(1)}, n_{\min}^{(1)} + \frac{1}{2} n_{\min}^{(1)} \log(\tilde{\lambda}) \right) \right]$$

where $n_{\min}^{(1)}$ is a dataset-specific minimum superpoints size for $\tilde{\lambda} = 1$. For example, for $n_{\min}^{(1)} = 50$, the smallest superpoint allowed for a small regularization strength $\tilde{\lambda} = 0.2$ will be 33, while it is 70 for the coarse partition obtained with $\tilde{\lambda} = 6$. While specific applications may require setting up this variable manually, this allowed us to produce the regularization paths in Figure 4 while only varying $\tilde{\lambda}$.

Optimization: Given the small size of our network, we train it for a short number of epochs (see Table 1), with decay events set at 0.7. We use Adam optimizer (?) with gradient clipping at 1 (?). Training takes around 2 hours per fold on our 11GB VRAM 1080Ti GPU.

Mini-batches: For graph-based clustering, the training phase processes batches of 16 point clouds at once, for which a subgraph of size 10 000 points is extracted. For the clustering-based segmentation, which is more memory intensive, and since subgraphs have to be larger to be meaningfully covered by the initial voxels, we set a batch size of 1 and a subgraph of 100 000. As a consequence, we

replace the batchnorm layers of the LPEs by group norms with 4 groups (?).

Augmentation: In order to build more robust networks, we added Gaussian noise of deviation 0.03 clamped at 0.1 on the normalized position and color of neighborhood clouds. We also added random rotation of the input clouds for the network to learn rotation invariance. To preserve orientation information, the clouds are rotated as a whole instead of each neighborhood. This allows the spatial transform to detect change in orientation, which can be used to detect borders.