Point Cloud Oversegmentation with Graph-Structured Deep Metric Learning (SUPPLEMENTARY)

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

1. Models configuration

In this section, we give the full hyper-parameterization of all the networks used in the paper, for both oversegmentation and semantic segmentation tasks, and for both datasets.

1.1. Models configuration for oversegmentation

Our supervized 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' border. 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 [1] (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 3.2.2 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 $c = \mid E \mid / \mid V \mid$ 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 3.2.1 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 $f^\star$ of (8) 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 7, we first train the network with a regularization strength of $\tilde{\lambda} = 1$ (see 3.2.2). 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:

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

where $n_{\text{min}}^{(1)}$ is a dataset-specific minimum superpoints size for $\tilde{\lambda} = 1$. For example, for $n_{\text{min}}^{(1)} = 50$, 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 7 while only varying $\tilde{\lambda}$.

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.

parameter	**S3DIS**	**vKITTI**
# parameters	278,897	118,737
Superpoint embedders configuration	[[64,64,128,128,256], [256,64,32]]	[[64,64,128,256], [128,32,32]]
STN configuration	[[64,64,128], [128,64]]	[[32,32,64], [64,32]
subsampling hops	4	
max SPgraph size	768	
$\tilde{\lambda}$	0.1	0.5
n_{min}	25	15
epochs	350	100
decay event	180,250,280,320	40,50,60,70,80

Table 2: Configuration of the semantic segmentation network. All values not mentioned in this table use default parameters from [6]

Optimization: Given the small size of our network, we train our network for a short number of epochs (see Table 1), with decay events set at 0.7. We use Adam optimizer [5] with gradient clipping at 1 [4]. 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.

1.2. Models configuration for semantic segmentation

We used the open-source superpoint-graph implementation `github/loicland/superpoint-graph` without any modification beyond changing the oversegmentation step and some changes in the hyper-parameters. The full parameterization is given in Table 2.

To compensate for the edges missed by the ℓ_0-cut pursuit approximation, due in part to its ignoring the spherical nature of the embeddings, we set the regularization strength $\tilde{\lambda}$ lower than 1 for both datasets. This help improve the accuracy and border recall. The subsequent decrease in border precision is compensated by the fact that the SPG, through its context leveraging module, can learn to propagate the semantic information to small superpoints. For the same reason, we chose a lower superpoint size for S3DIS from the segmentation experiments.

We extended the superpoint graph subsampling threshold to 4-hops instead of 3, because our method SSP tends to produce thin components near interfaces. Since the vKITTI

Method	OA	mAcc	mIoU	ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter
A5 PointNet [8]	–	49.0	41.1	88.8	97.3	69.8	**0.1**	3.9	46.3	10.8	52.6	58.9	40.3	5.9	26.4	33.2
A5 SEGCloud [9]	–	57.4	48.9	90.1	96.1	69.9	0.0	18.4	38.4	23.1	75.9	70.4	58.4	40.9	13.0	41.6
A5 PointCNN [7]	85.9	63.9	57.3	**92.3**	**98.2**	79.4	0.0	17.6	22.8	62.1	80.6	74.4	66.7	31.7	**62.2**	**56.7**
A5 SPG [6]	86.4	66.5	58.0	89.4	96.9	78.1	0.0	**42.8**	48.9	**61.6**	84.7	75.4	69.8	**52.6**	2.1	52.2
A5 SSP + SPG (ours)	**87.9**	**68.2**	**61.7**	91.9	96.7	**80.8**	**0.0**	28.8	**60.3**	57.2	**85.5**	**76.4**	**70.5**	49.1	51.6	53.3
PointNet [8] in [2]	78.5	66.2	47.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	42.0	54.1	38.2	9.6	29.4	35.2
Engelmann *et al.* [2]	81.1	66.4	49.7	90.3	92.1	67.9	44.7	24.2	52.3	51.2	47.4	58.1	39.0	6.9	30.0	41.9
Engelamnn in [3]	84.0	67.8	58.3	92.1	90.4	78.5	37.8	35.7	51.2	65.4	61.6	64.0	51.6	25.6	49.9	53.7
SPG [6]	85.5	73.0	62.1	89.9	95.1	76.4	62.8	47.1	55.3	68.4	73.5	69.2	63.2	45.9	8.7	52.9
PointCNN [7]	**88.1**	75.6	65.4	**94.8**	**97.3**	75.8	**63.3**	51.7	58.4	57.2	69.1	**71.6**	61.2	39.1	52.2	58.6
SSP + SPG (ours)	87.9	**78.3**	**68.4**	91.7	95.5	**80.8**	62.2	**54.9**	**58.8**	**68.4**	**78.4**	69.2	**64.3**	**52.0**	**54.2**	**59.2**

Table 3: Results on the S3DIS dataset on fold "Area 5" (top) and micro-averaged over all 6 folds (bottom). Intersection over union is shown split per class, with the highest value over all methods in bold.

dataset is much smaller than S3DIS, we chose smaller networks to mitigate overfitting.

2. Residual Point Embedder

We have tested an alternative configuration for the local point embedded, in which they were stacked in layers, similarly to the classical convolutional architecture for images. We first introduce a slightly changed architecture, the Residual Point Embedder RPE, whose design is based on an LPE but takes a supplementary input e_{ini}. Instead of computing a new embedding, the RPE computes a residual (1) which is added to this initial embedding before normalization (2):

$$R(X_i, x_i) = \text{MLP}_2\left(\left[\max\left(\text{MLP}_1(X_i)\right), x_i\right]\right) \quad (1)$$
$$\text{RPE}(x_i, X_i, e_{\text{ini}}) = \text{L}_2\left(e_{\text{ini}} + R(X_i, x_i)\right) \quad (2)$$

The second change is the layers architecture. The RPEs in the first layer compute the embeddings from the local geometric and radiometric information alone, and their initial embedding is set to 0 (3) (such that they behave exactly like LPEs). The RPEs in subsequent layers compute new embeddings from the local radiometry and geometry as well as the embeddings computed at the previous layer of the points neighbors E_i^t (4). Note that for a point to be processed by a layer, all its neighbors must have been embedded by the previous layer. This allows the RPEs to have increasingly broader receptive fields, and to correct errors that might have been done by previous layers. Note that the geometric information are only processed by the spatial transform once, cascading its values to all residual layers.

$$e_i^{(0)} = \text{RPE}^{(0)}([\tilde{P}_i, R_i], [\tilde{p}_i, r_i], 0) \quad (3)$$
$$e_i^{(t+1)} = \text{RPE}^{(t)}([\tilde{P}_i, E_i^{(t)}], [\tilde{p}_i, r_i, e_i^{(t)}], e_i^{(t)}) \quad (4)$$

Alternatively, all initial embeddings can be set to 0, which means that each layer computes a new embedding from the local position and the embeddings of the previous layers. As mentioned in the ablation study, while these networks did perform well, their benefits shrink when a simple LPE is given as many parameters.

3. Detailed results and illustration

We present in Table 3 the per-class IoU for the S3DIS dataset. We illustrate the semantic segmentation results in Figure 1. We also made a video illustration which can be accessed at https://youtu.be/bKxU03tjLJ4.

References

[1] B. Delaunay et al. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2, 1934. 1

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

[3] F. Engelmann, T. Kontogianni, J. Schult, and B. Leibe. Know what your neighbors do: 3d semantic segmentation of point clouds. *arXiv preprint arXiv:1810.01151*, 2018. 3

[4] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016. 2

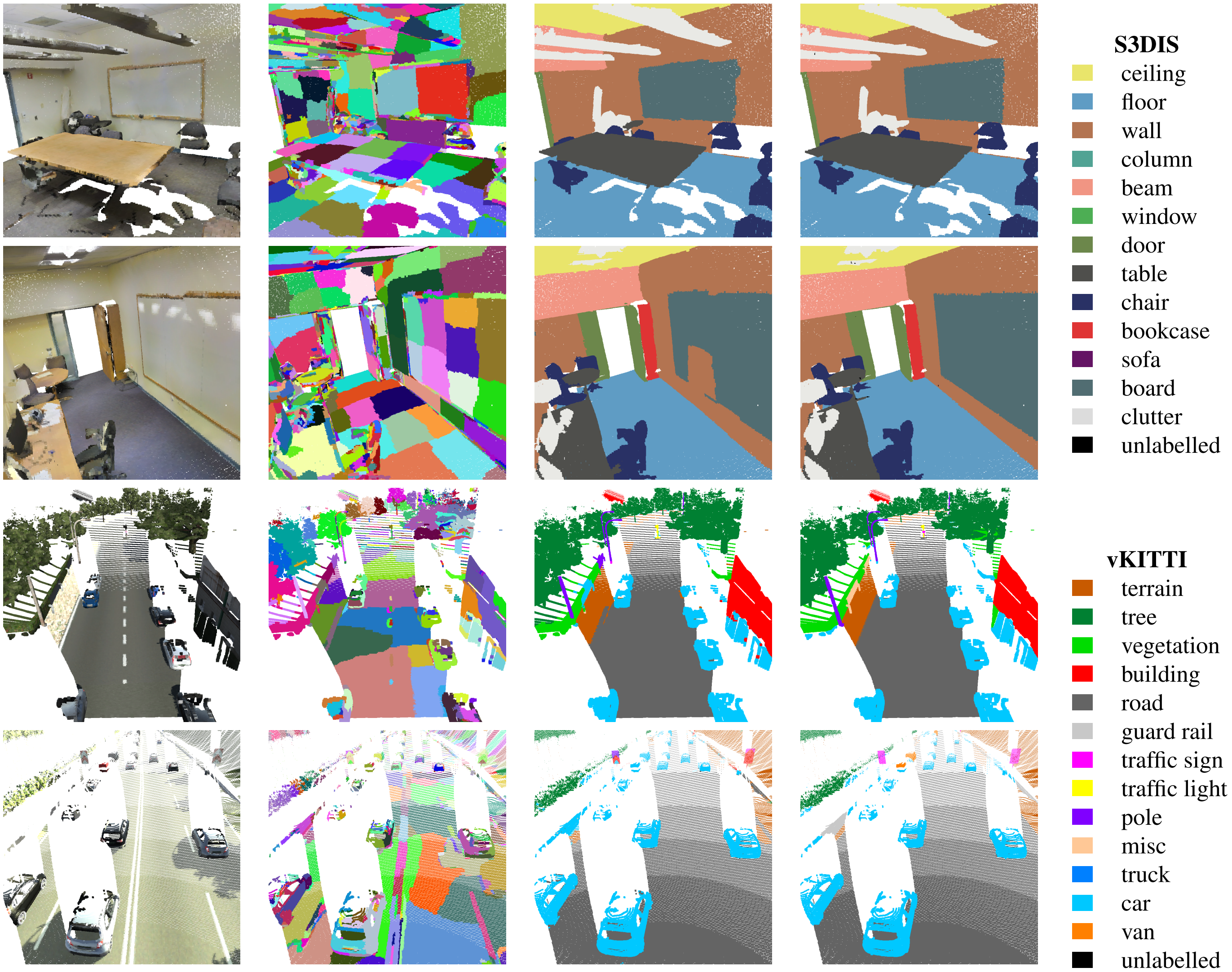
[5] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 2

[6] L. Landrieu and M. Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*. IEEE, 2018. 2, 3

[7] Y. Li, R. Bu, M. Sun, and B. Chen. PointCNN. *arXiv preprint arXiv:1801.07791*, 2018. 3

[8] 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. 3

[9] 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. 3



(a) Input Point Cloud (b) Oversegmentation (c) Semantization (d) ground Truth

Figure 1: Illustration of the results on the semantic segmentation. In the first row we show a successful semantization for a complex scene of S3DIS. In the second row, we show a failure case in which a white board is oversegmented in too many small superpoints. This makes their classification harder by the semantic segmentation network. In the third row we see a successful semantization of an urban outdoor scene from vKITTI. On the fourth row, we can observe in the background road signs with high color contrasts, which are segmented in small superpoints. This makes them very hard to classify and they are missed by the semantic segmentation algorithm.