RAILWAY LIDAR SEMANTIC SEGMENTATION WITH AXIALLY SYMMETRICAL CONVOLUTIONAL LEARNING

Antoine Manier[1,2*], Julien Moras[2], Jean-Christophe Michelin[1], Hélène Piet-Lahanier[2]

[1]SNCF Réseau – DGII TTD, 9 Avenue François Mitterand, 93210 Saint-Denis, France
[2]ONERA – DTIS, 6 chemin de la Vauve aux Granges, 91120 Palaiseau, France
[*]manier.antoine@gmail.com
jean-christophe.michelin@reseau.sncf.fr
{julien.moras / helene.piet-lahanier}@onera.fr

Commission II, WG II/3

KEY WORDS: Semantic segmentation, 3D point cloud, Deep-learning, Railway, LIDAR

ABSTRACT:

This paper presents a new deep-learning-based method for 3D Point Cloud Semantic Segmentation specifically designed for processing real-world LIDAR railway scenes. The new approach relies on the use of spatial local point cloud transformations for convolutional learning. These transformations allow an increased robustness to varying point cloud densities while preserving metric information and a sufficient descriptive ability. The resulting performances are illustrated with results on railway data from two distinct LIDAR point cloud datasets acquired in industrial settings. The quality of the extraction of useful information for maintenance operations and topological analysis is pointed together with a noticeable robustness to point cloud variations in distribution and point redundancy.

1. INTRODUCTION

3D data are becoming a standard for environment perception, and replace images in various use cases: autonomous cars, urban mapping, forensics. They usually come as point clouds (coordinates in the 3D space) with an associated radiometry information. As deep-learning for processing 2D images has already developed towards industrial use (face recognition, CGI, cartography), deep-learning for processing 3D point clouds is an ongoing field of research. In the past few years, the autonomous cars industry has been a driving force of its development especially by providing open source data – e.g. SemanticKITTI (Behley et al., 2019).

With the uprising embeddability and decreasing financial cost of high precision devices (LIght Detection And Ranging scanners – LIDAR) point clouds data for geomatics and large scale topography become widely accessible (e.g. the OpenTopography[1] database). Several large scale high density open source LIDAR datasets – e.g. Hessigheim Benchmark (Kölle et al., 2021) – address the issue of Point Cloud Semantic Segmentation (PCSS) i.e. labeling each point with relevant semantic information. Although this information is low-level, it is used to build meaningful representations for real-world application – e.g. numerical terrain models, vegetation mass estimate, population density, infrastructures cartography – see the work of Soilán et al. (2019) for a comprehensive review in the railway field. Granting the railroad industry has been using LIDAR data for a while, especially for these type of applications, deep learning-based processing is yet at its early stages of development in this field. The use of reliable point cloud semantic information could simplify existing railway use cases and lead to new ones (vegetation growth estimate, automated site inventory, assisted building information modeling). Railway LIDAR PCSS represents a first step towards optimizing maintenance.

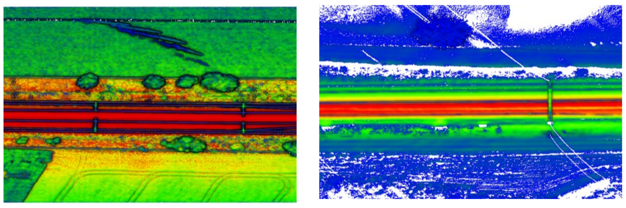
Figure 1. Top view of railway LIDAR point clouds using: airborne capture (left), railborne capture (right), colorized by relative local density (low-medium-high / blue-green-red)

In this paper we address the issue of semantic segmentation on large scale railway LIDAR point clouds by proposing a simple partially symmetrical kernel operator supported by a deep-learning framework. We design our method to be specifically adapted to the characteristics of railway LIDAR data. Evaluation of the resulting approach is performed using railway data from two distinct LIDAR point cloud datasets. The paper is organized as follows: we first introduce the railway context and our datasets as to identify their specificity. We then discuss several PCSS methods relevant to the specifics of our use case. We present our deep-learning approach designed for railway environment then evaluate our results of segmentation, speed and robustness to point distribution together with several state-of-the-art methods. We finally elaborate on the future developments our approach could benefit from.

2. RAILWAY CONTEXT

SNCF Reseau, the french railway state-owned company, coordinates the maintenance operations of $\sim$27,000km of high speed and regional active railway lines. Its numerical twin project aims at modeling the entire network and its infrastructure,

* Corresponding author
[1] https://opentopography.org

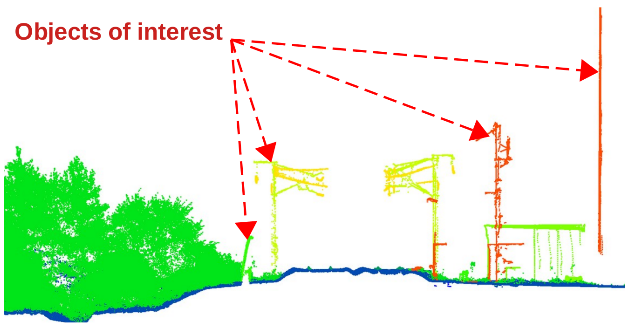

Figure 2. Slice from an aerial LIDAR point cloud of railway tracks colorized by semantic class

centralizing positional and semantic information along with any maintenance-related details into a unique model. It is part of an ongoing effort to optimize existing operations and evolve towards predictive maintenance.

To collect high precision topological data on a regular basis, SNCF Reseau has been using railborne and airborne LIDAR systems for the past few years. To keep information up to date, data acquisition campaigns are continuously led throughout the entire network hence providing a significant and still increasing amount of available data to process. The resulting 3D point clouds (see Figure 1) are currently being used to perform automated analyses with model-based algorithms for track vectorization, vegetation to infrastructures proximity warnings or catenary pole geolocation. While focusing on specific infrastructure objects (see Figure 2), we want to have a better estimate of the benefits of deep-learning PCSS in railway environments.

To compete with existing model-based industrial approaches we want to achieve a good quality of segmentation (as it is the very first step to maintenance related object extraction) while keeping a low processing time – the amount of collected data being consequent. Also, since the ground truth data we present in Section 2.1 is not definitive and will be updated (possibly on a regular basis) to be more representative of the overall infrastructures, we want our deep-learning based solution to be fast to train.

2.1 Railway datasets

To tackle our use case, we introduce two datasets acquired on French railroads, annotated for semantic segmentation evaluation purposes (see Table 1). The former (Paris-Lyon) is a hand-labeled (high semantic quality) airborne LIDAR point cloud dataset of 5.25km of linear tracks on a high-speed line. The later (Saint-Etienne) is a semi-automatically labeled (lower semantic precision) railborne LIDAR set of 13km on a regional line.

Dataset	Type	Length (km)	N_{pts} (M)	Seg. truth
Paris-Lyon	Airborne	5.25	297.5	Manual
Saint-Etienne	Railborne	13	257.0	Semi-auto

Table 1. Railway LIDAR datasets characteristics

The environment in the Saint-Etienne is much more diverse than in Paris-Lyon. Both are labeled using the same generic set of classes: $Ground$, $Rail$, $Vegetation$, $Fence$, $Building$, $CatenaryPole$, $CatenaryWire$, $Structures$, $Environment$. While $Structures$ encompasses rail bridges, road bridges and tunnels, $Environment$ contains all remaining railway related objects (beacons, antennas, electrical cabinet, etc.) and punctual objects (lightning pole, high voltage lines, cars, etc.). As we focus mainly on the railway infrastructure, we keep a reduced number of classes to facilitate our PCSS problem while still segmenting out important objects. Both datasets contain the same information per point collected from the LIDAR scanner: XYZ the 3D spatial coordinates, $Intensity$ of the returning signal, N_{ret} the number of returns per laser beam and R_{id} the return index in a series of returns per beam.

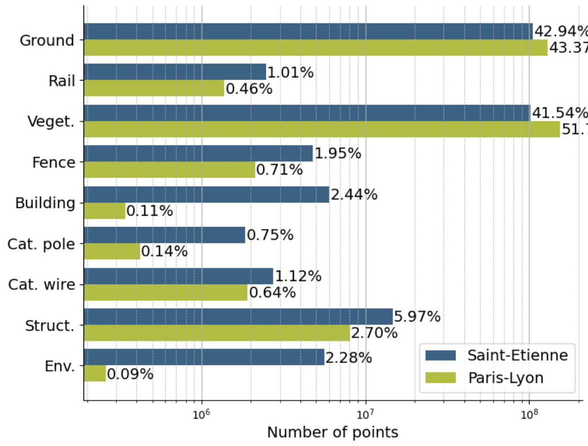

Figure 3. Number of points per class (and percentages) for each of the railway datasets

2.2 Analysis

Here we want to identify the characteristics of our datasets in the scope of our PCSS task. These characteristics will orient our choices in the development of our specific approach. As we focus on railway infrastructure, we notice in Figure 2 that objects of interest are mainly oriented vertically. The ground beneath the tracks being necessarily flat for complying with railroad standards, objects like antennas, catenary poles, signalization poles or electrical cabinets always show a major Z-oriented component.

A usual inconvenient of the LIDAR scanning approach is the locally varying distribution of points in the resulting cloud. As we notice in Figure 1, areas close to the sensor bear more point density than those away from it. We also notice that a lower point of view (railborne) leads to more local density variations than a higher point of view (airborne). Finally, as objects of interest are very scarce in both datasets, we struggle with a strong class imbalance (see Figure 3). Saint-Etienne dataset shows a greater proportion of points from several misrepresented classes than Paris-Lyon.

3. RELATED WORKS

Keeping in mind our objectives of segmentation quality, training and inference speed and robustness to varying density, we review in this section a few state-of-the-art trends and techniques for PCSS relevant to our use case.

3.1 Railway LIDAR PCSS

In our specific industrial context, traditional model-based approaches are often object oriented and rely on user-implemented geometrical global and local features, trajectory

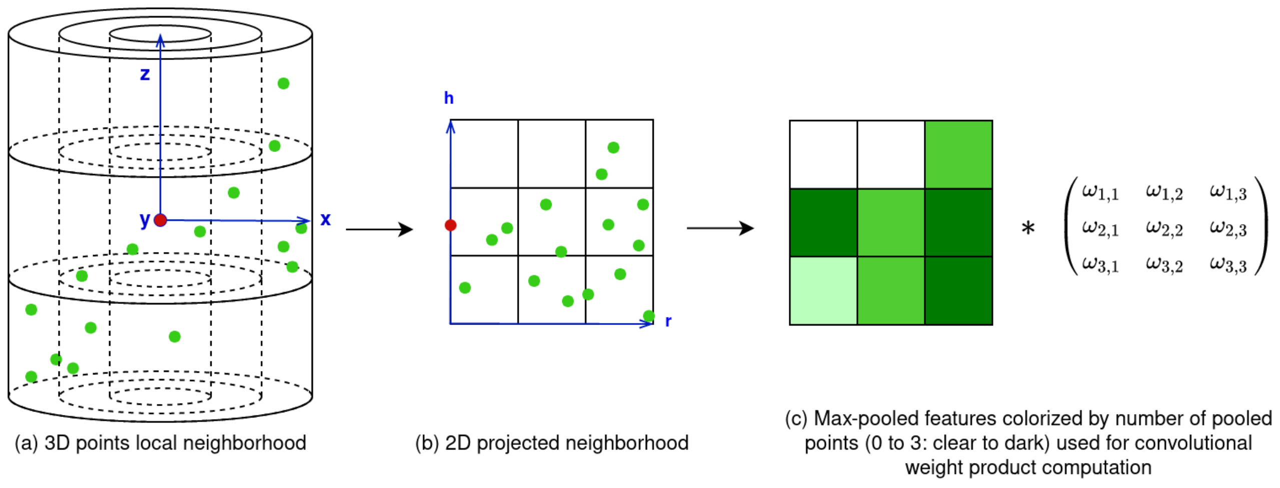
z
y
x
h
r
$* \begin{pmatrix} \omega_{1,1} & \omega_{1,2} & \omega_{1,3} \\ \omega_{2,1} & \omega_{2,2} & \omega_{2,3} \\ \omega_{3,1} & \omega_{3,2} & \omega_{3,3} \end{pmatrix}$
(a) 3D points local neighborhood
(b) 2D projected neighborhood
(c) Max-pooled features colorized by number of pooled points (0 to 3: clear to dark) used for convolutional weight product computation

Figure 4. Successive steps for performing local projective cylindrical convolution with: (a) the 3D cylindrical neighborhood (green) centered around the query point (red); (b) the spatial 2D projection and partitioning of the neighborhood; (c) the max-pooling of the neighborhood points' attached spectral features into a 3 × 3 grid used as a convolution support for the weight matrix $\Omega^{3\times3}$

information – e.g. Lamas et al. (2021). Although these approaches perform well in controlled environments, they need to be redesigned for any change in use case and are sensitive to unforeseen characteristics.

In recent work, Guinard et al. (2021) propose a fast Random forest classifier using a series of handcrafted features derived from eigenvalues of local point-neighborhoods covariance matrices. To improve the algorithm's robustness to ill-sampled data and processing speed, the authors design a pre-segmentation step of the point cloud into geometrically homogeneous, simple segments. Classification is performed on those segments and propagated to the original points. The methods shows very good results although on a very small dataset.

Exploring deep-learning approaches, Soilán et al. (2020) apply PointNet (Qi et al., 2017a) and KPCNN (Thomas et al., 2019) to segmentation of railway tunnels point cloud data. Although the test environment is simple, their work shows hopeful results for full infrastructure segmentation using deep-learning-based approaches.

3.2 Deep-Learning for LIDAR PCSS

A major issue in deep-learning PCSS lies in the definition of the data: as opposed to 2D pixels or 3D voxels images, point clouds are unstructured thus disabling straightforward discrete convolutional operations. Several approaches work around this issue using intermediate global representations, i.e. projecting point clouds onto 2D grids (Boulch et al., 2018) or 3D voxels (Tchapmi et al., 2017).

PointNet (Qi et al., 2017a) stands as a pioneer in proposing to work directly on unstructured point cloud. Rich features learning without the need of a proxy representation of the point cloud is achieved by approximating n-dimensional global filters with Multi-Layers Perceptrons (MLP). Points coordinates are directly passed as features for learning. Since MLP layers process points individually, the local context is aggregated through pooling operations. Although lacking the descriptive power of a local operator, this approach proves to be efficient and extremely modular.

Recent works (Boulch, 2020; Thomas et al., 2019) introduce a new type of continuous 3D local convolution using an unstructured kernel allowing a fine contribution from each local neighborhood point to the convolution product. The downside to this approach is its sensitivity to data redundancy – e.g., two spatially close points bearing similar spectral information will both contribute equally to the convolutional product.

In a more straightforward design, some methods propose a structured discrete kernel as 3D local voxels (Hua et al., 2018) or 1D bins (Zhang et al., 2019). Though both approaches operate on a point's local neighborhood (thus preserving the point cloud global representation) they imply a spatial projection while calculating the learned features. At the risk of limiting kernel descriptiveness, the ShellConv kernel (Zhang et al., 2019) keeps a low number of learned parameters and speeds up learning and inference.

4. A PARTIALLY SYMMETRICAL KERNEL FOR LOCAL POINT CLOUD DESCRIPTION

To fulfill our objectives of segmentation, speed and robustness to varying density, we inspire from the ShellNet projective descriptor (Zhang et al., 2019). Its kernel design is simple but efficient and allows a significant increase in inference speed. To benefit from the seemingly important elevation information of our objects of interest (cf. Figure 2), we design a vertically oriented cylindrical kernel operating on local point neighborhood. Working with metric data, we also want to preserve metric information along the transformations as it can be discriminant (e.g. between a shack and an electrical cabinet).

4.1 A focus on neighborhood selection

As the calculation of the convolution product is to be performed on a local neighborhood of points (see Figure 4, a), a point selection operation must be used to define this neighborhood. Several recent works (Zhang et al., 2019; Thomas et al., 2019; Boulch, 2020) use a K-Nearest Neighbors algorithm (K-NN) and normalize the selected points to a unit sphere. Although easily parallelizable, this approach neither guarantees a fixed size neighborhood nor metric conservation hence overlooking potentially discriminant tacit information.

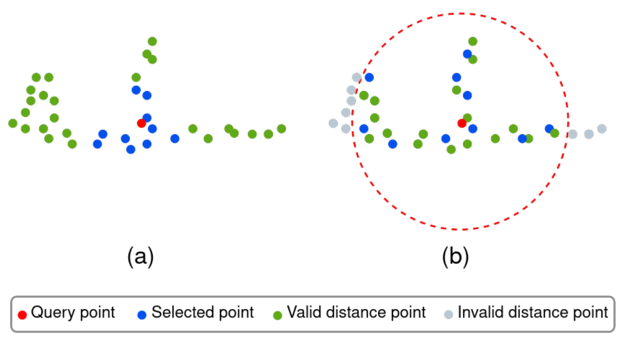
(a)
(b)
● Query point ● Selected point ● Valid distance point ● Invalid distance point

Figure 5. Schema of a K-NN selection (a) and a fixed distance neighborhood selection (b)

An alternative approach is proposed here. Define a sphere of radius r_s centered on a query point p. Let $\mathcal{S}$ be the set of all neighboring points within this sphere, i.e. $q \in \mathcal{S} \Leftrightarrow ||p - q|| < r_s$. The proposed approach samples uniformly K points from $\mathcal{S}$ (see Figure 5). This sampling approach is efficiently implemented in the python package ”Torch-Cluster” (Fey, 2020). We find that this implementation allows the selection of a large number of neighbors without significant impact on the overall algorithm speed, contrary to a K-NN search. Moreover, the fixed-size approach seems to help increasing robustness to local variations in local points distribution (cf. Section 5.3).

In the case where $K < |\mathcal{S}|$, drawn points are repeated which does not influence the convolution product thanks to the spatial binning and max-pooling operations later performed (see Section 4.3). To avoid under-sampling, i.e. $K \ll |\mathcal{S}|$ or overly repeating information i.e. $K \gg |\mathcal{S}|$, we aim at selecting r_s such as $K \approx |\mathcal{S}|$. At each layer's first forward pass, for each input point p, we use a K-NN algorithm to calculate $d_p = \max_{\forall q \in N_p^k} ||p - q||$. The averaged maximal distance per K-neighborhood over the input point set P is defined as $r_s = \frac{1}{|P|} \sum_{p \in P} d_p$. Once r_s is fixed for each layer, the training process only uses the distance search.

4.2 An axially symmetrical projection

One of the main objectives of this work is to take advantage of the inherent characteristics of the railway scenes – i.e. objects of interest have a strong vertical component. Although ShellNet (Zhang, et al., 2019) uses a fully invariant descriptor, we define an axially symmetrical transformation to conserve relative elevation information. This transformation is applied around the query point $p = \begin{bmatrix} x_p & y_p & z_p \end{bmatrix}^T$, with $p \in N_p^k$ the K-neighborhood of p obtained using the approach described in Section 4.1. It consists in projecting the K-neighboring 3D points $q \in N_p^k$ into 2D space using the transformation Φ expressed in Equation 1. Where x, y, z are the points 3D coordinates, z_{min} denoting the minimum z value of the neighborhood points N_p^k.

$$\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$$

$$\Phi_p(q) = \begin{bmatrix} ||(x_q - x_p, y_q - y_p)|| \\ z_q - z_{min} \end{bmatrix} \text{ where } z_{min} = \min_{\forall q \in N_p^k} z_q \tag{1}$$

The resulting 2D neighborhood is illustrated in Figure 4, b. Although this transformation simplifies the K-neighborhood expression, a downside to this strategy is the loss of information due to projecting 3D data onto 2D domain, even while preserving elevation, possibly impeding on the kernel overall descriptiveness.

4.3 Binning and max-pooling operation

Because spinning LIDAR sensors sample regularly in angular domain, points are sampled unevenly in height/range space depending of their distance to the sensor. To deal with this issue, Zhu et al. (2021) propose to use a cylindrical spatial partitioning centered on the sensor for processing single scans. In this geometry, scans are transformed from cartesian to cylindrical coordinate system and projected into a cylindrical grid. Doing so, the number of points in cells become roughly independent of their distance to the sensor. Although this idea is interesting to deal with sampling issue, it is restricted to process scan one at a time (or at least implies having information about the sensor location over time) since the absolute geometry changes between scan acquisitions. In our use case, scans are already concatenated into dense point clouds thus presenting a different density profile illustrated in Figure 1. Moreover, in our trainborne and airborne configurations, the embedded spinning LIDAR sensors are single-sheet and scanning perpendicularly to the movement vector. Nonetheless, this approach can be used to regularize density with a local statistical approximation instead of global one.

A spatial K-neighborhood points grouping within bins is then performed as shown in Figure 4, b. For the point p its feature vector at the output of the operator $F^{out}(p)$ is computed from feature vectors of the the N_p^k input points $F^{in}(q)$ using Equation 2. For each bin B_i, the feature vector of all point belonging to B_i are merged using a $MaxPooling$ operator. The resulting feature map (one feature vector for each bin) is then convoluted with the filter (parametrized by the weights ω_i) (see Figure 4, c).

$$F^{(n)}(p) = \sum_{i}^{|B|} \omega_i \cdot MaxPool(\{F^{(n-1)}(q) : \forall q \in B_i\}) \tag{2}$$

As the binning operation obviously reduces the intrinsic measure-induced noise or local variations in points distribution, the max-pooling operator (see Equation 2) provides a regularization by skipping redundant local information – e.g., two spatially close points bearing similar spectral information and falling into the same bin contribute only once to the convolutional product.

5. RESULTS

In this part we first introduce the network architecture used to support the convolutional operator. We then present the method's results and compare its segmentation performances, execution speed and robustness to varying density to other state-of-the-art approaches.

5.1 Network design

The proposed approach is based on the well known U-Net (Ronneberger et al., 2015) (encoder-decoder) architecture for segmentation used in many PCSS methods (Tchapmi et al., 2017; Boulch, 2020; Thomas et al., 2019), where 2D convolutional

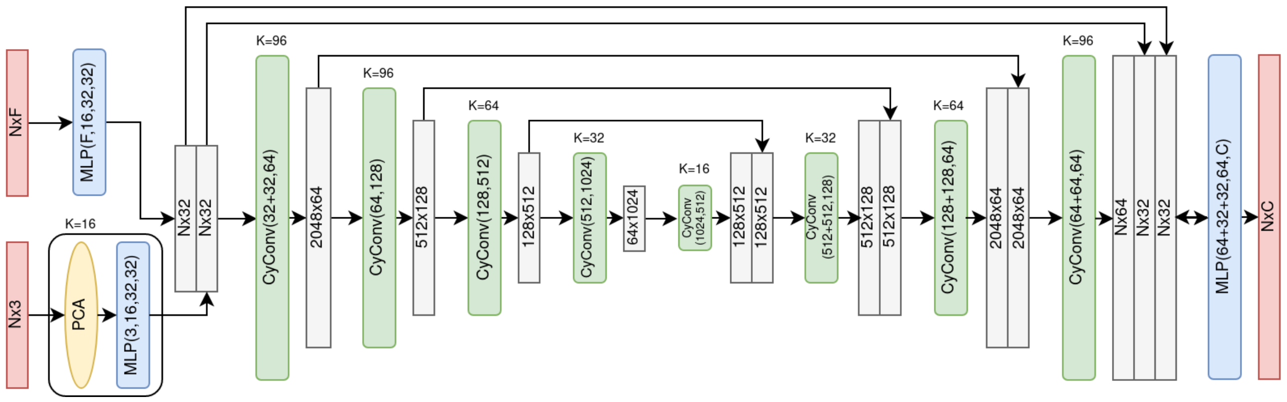

Figure 6. Structure of the encoder-decoder network with the Cylindrical convolutions, the MLP and PCA modules. Each CyConv layer also takes as input the downsampled point cloud along with the previous layer's features in order to perform the spatial pooling. K denotes the numbers of searched neighbors per input point at each layer level.

blocks are replaced with the cylindrical 3D convolution proposed in Section 4. This model is illustrated in Figure 6. The successive dimensional reductions, here computed using Farthest Point Sampling (FPS) algorithm (as introduced by Qi et al. (2017b)), along the steps of the network reduce computational time and memory needs. The skip connections between the encoder and the decoder allow a proper update of the first layers's weights where the spatial information is being encoded – an essential step especially with 3D points clouds holding few initial spectral information. Spectral (NxF) and spatial information (Nx3) are processed by two separated frontend before feeding the convolutional encoder-decoder.

The spatial frontend is composed of a Principal Component Analysis (PCA) module followed by a shared MLP and operates as follows: for each input point p_{xyz} its k spatial neighbors P_k are uniformly selected within a fixed distance sphere (cf. Figure 5 and Section 4.1). The covariance matrix of the centered neighborhood's positions $C = \frac{P_k^T \times P_k}{k-1}$ is diagonalized such as $C = V \times \Sigma \times V^T$ where V is the matrix of eigenvectors, Σ the diagonal matrix holding the eigenvalues $\lambda_1 \leq \lambda_2 \leq \lambda_3$ in decreasing order. Once normalized, these eigenvalues are used as rotationally invariant local features and lifted to higher dimension using a 3-layers shared MLP composed of respectively 16, 32 and 32 nodes. The spectral frontend is composed of a single 3-layers shared MLP composed of respectively 16, 32 and 32 nodes applied to the LIDAR features $Intensity$, N_{ret} and R_{id} introduced in Section 2.1.

Both resulting spatial and spectral features are stacked together and fed into the convolutional backbone consisting in 8 convolutional layers. Finally, the last MLP layer decodes the output features to the desired output semantic classes.

5.2 Analysis of semantic segmentation quality

This part presents the segmentation performances of the proposed method and compares them to the similar state-of-the-art methods ConvPoint (Boulch, 2020) and ShellNet (Zhang et al., 2019) over the two railway datasets presented in Section 2.1. Segmentation performances were evaluated using the widely used mean Intersection-over-Union (mIoU) metric as described by Qi et al. (2017a).

For fair comparison, the learning parameters of our approach have been chosen to be close to those originally used in the tested methods. To select the network inputs, the dataloader proceeds as follows: 8 points are randomly selected from the entire training dataset, used as centers for column-shaped areas of $10m \times 10m$ wide. All points within those 8 areas are each randomly sampled to 8192 points and concatenated as input – in areas containing less than 8192 points, samples are repeated. Each point is represented by its XYZ position and its base LIDAR information introduced in Section 2.2 as normalized and centered input features. The inference split of Paris-Lyon dataset represents 14.3% of the entire dataset (750m) and 19.2% for Saint-Etienne (2.5km).

The network is trained using a cross-entropy loss function, an initial learning rate $L_0 = 1 \times 10^{-3}$ along with an Adam optimizer. Each epoch is composed of 100 iterations over the dataloader, i.e. $100 \times 8 \times 8192$ points. The current learning rate is decreased every 50 epoch by a factor 0.9. All methods use a dropout parameter of 0.5 and batch normalization steps while training. As both datasets' classes are unbalanced (see Figure 3), a loss weighting coefficient is applied during training: $\rho = \sqrt[2.5]{\left(\frac{P_{max}}{P_c}\right)}$ $\forall c \in C$ the classes, with P the percentage of a class in points within the dataset. It allows a stronger error back-propagation for the less represented classes.

Semantic segmentation performance results are summarized in Table 2. The proposed method shows comparable overall results with both tested state-of-the-art methods. It systematically outperforms both methods while segmenting large objects like buildings, concrete structures, vertical objects like catenary poles or environment objects (encompassing antennas, side-track beacons, signal poles, electrical cabinets or power-line poles) but, mostly underperforms while segmenting the ground, rails or vegetation (i.e. classes without a dominant vertical component).

We also notice that the strong class imbalance does not necessarily impact the segmentation performances. In Paris-Lyon results, all methods perform properly on the catenary poles class but not on the building class (respectively 0.14% and 0.11% of the dataset's points). ShellNet fails to segment the building class altogether. This result can be explained by the regular spatial distribution of catenary poles within the point clouds, contrary to the buildings. The batches being drawn randomly across the whole dataset, the building class is not seen often during training. We can conclude that in this configura-

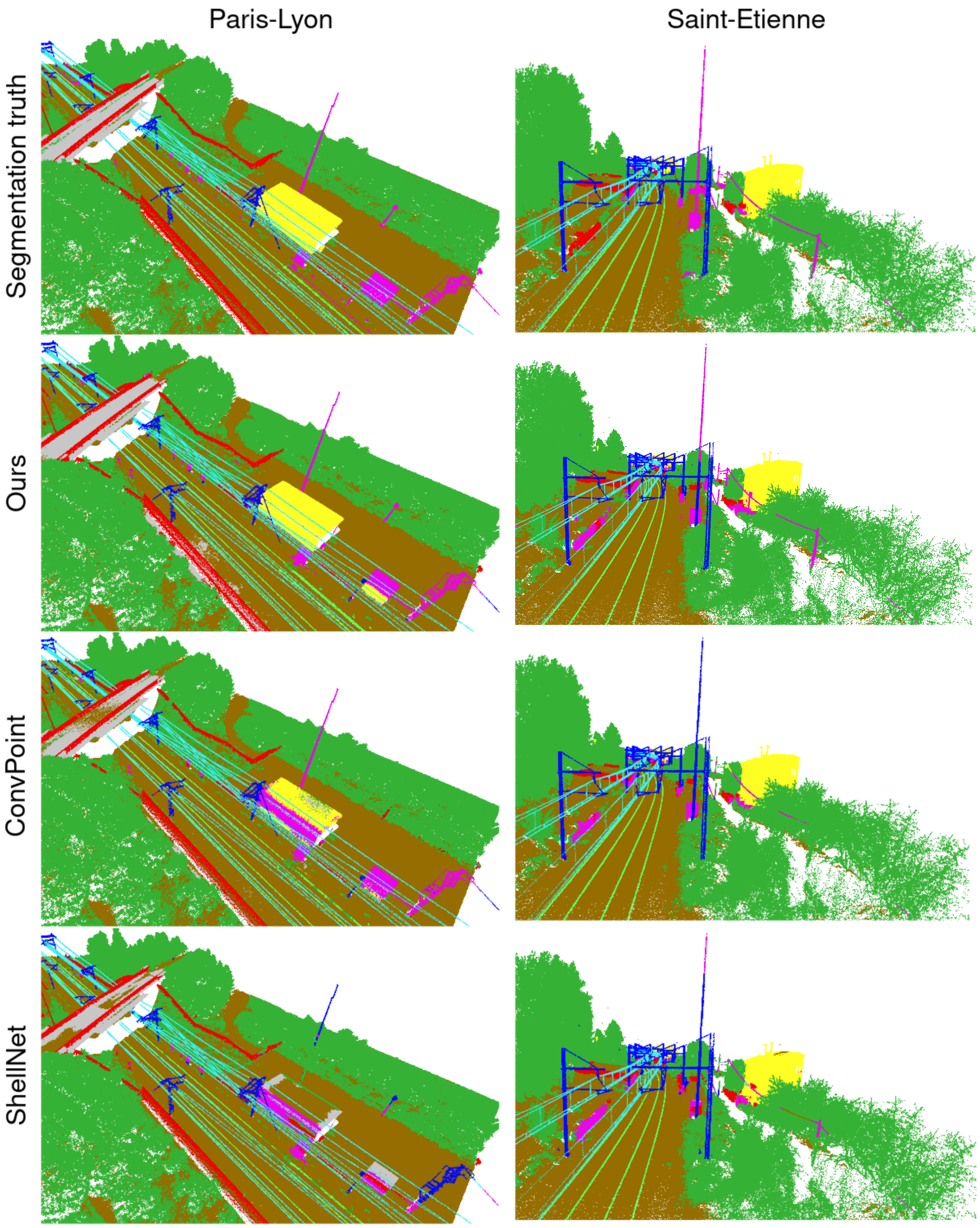

Figure 7. Colorized segmentation results on both railway datasets with: $Ground$ in brown, $Rail$ in bright green, $Vegetation$ in plain green, $Fence$ in red, $Building$ in yellow, $CatenaryPole$ in blue, $CatenaryWire$ in cyan, $Structures$ in grey, $Environment$ in bright pink.

Dataset	Method	mIoU	Ground	Rail	Vegetation	Fence	Building	Cat. Pole	Cat. Wire	Struct.	Env.
Paris-Lyon	ConvPoint	78.1	**88.1**	**94.5**	**87.7**	**84.5**	54.0	82.9	97.6	73.9	39.7
	ShellNet	66.1	86.0	90.7	85.8	78.8	0.0	73.0	97.3	68.9	14.5
	Ours	**83.4**	82.4	89.1	82.3	79.5	**97.2**	**86.1**	**98.0**	**84.7**	**51.8**
Saint-Etienne	ConvPoint	76.7	**95.8**	**84.5**	**96.9**	35.0	78.0	83.8	96.5	92.0	28.0
	ShellNet	73.3	93.3	78.4	95.1	32.2	77.7	79.1	93.2	91.2	20.2
	Ours	**77.3**	94.6	77.9	96.5	**36.9**	**81.2**	**90.1**	**96.8**	**92.1**	**29.7**

Table 2. Segmentation performances in mIoU (%) by class on Paris-Lyon and Saint-Etienne datasets

tion, the spatial distribution of objects within the dataset has a strong impact on the segmentation performances, especially in the case of statistically unbalanced dataset.

To better understand the caveats behind the metrics, we display comparative views of the resulting segmented points clouds in Figure 7. In Paris-Lyon, we see that both ShellNet and ConvPoint show confusion between the bridge (structure) and the ground class while our approach does not. This is most likely due to its large metric receptive field. As a downside, it presents a lower accuracy on the fine-grained textures like the low vegetation. In Saint-Etienne, we notice that our approach is more successful in segmenting the antenna which is attributed entirely to the catenary pole class with ConvPoint and partially with ShellNet. This aspect is most likely due to the tacit preservation of the metric information along the network. Our approach also seems to be less sensitive to spatial noise while segmenting objects inside the vegetation (environment pole and wire on the bottom right of the Saint-Etienne thumbnails).

5.3 Processing speed and robustness to point distribution

Although our segmentation results are competitive with the selected state-of-the-art approaches in a railway environment, the industrial constraints discussed in Section 2.2 lead to the matters of training and inference speed and robustness to variations in point cloud distribution.

Method	Paris-Lyon		Saint-Etienne	
	Train (min)	Inference (s)	Train (min)	Inference (s)
ShellNet	1067	1568	388	**1666**
ConvPoint	144	**1286**	**85**	2232
Ours	**94**	1475	375	2075

Table 3. Comparison of compute time for training phase and inference phase on both datasets (lower is better)

Training and inference times necessary to achieve the segmentation performances displayed in Table 2 are summarized in Table 3. In our inference settings, we chose for ShellNet and our method to sample a $10m \times 10m$ wide column every 2 meters, offering a satisfying compromise between speed and segmentation performances. As ConvPoint showed very low segmentation performances with a 2 meters step, we reverted to the author's original 0.8 meter step.

The proposed method converges faster (train) on Paris-Lyon ($\sim$10 times than ShellNet and $\sim$1.5 times than ConvPoint), and shows an inference time consistent with those of the other methods – although ConvPoint shows slightly better results. On Saint-Etienne however, the proposed method struggles to reach convergence within competitive time (ConvPoint is $\sim$4.5 times faster than the other methods), the inference times are similar for all methods – although slightly better for ShellNet. The Saint-Etienne dataset scene complexity being greater, smaller networks (ShellNet and ours) might suffer from a lack of abstraction compared to ConvPoint (embedding a 13-layers-deep network), leading to a slower convergence.

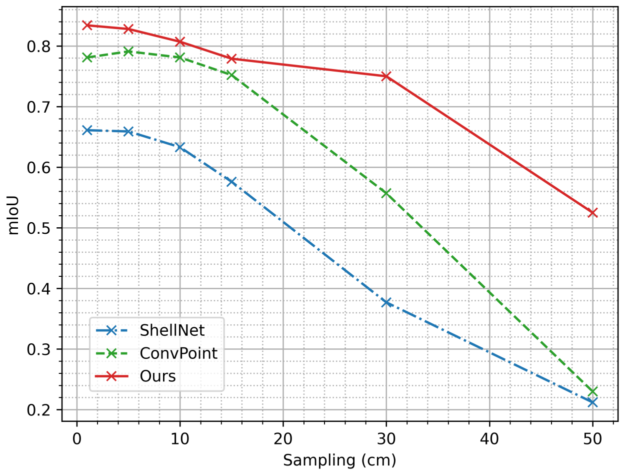

Figure 8. Effect of different grid-sampling resolutions with the Paris-Lyon inference split on the segmentation mIoU metric. The tested resolutions are 1cm, 5cm, 15cm, 30cm, 50cm.

An other aspect of our kernel approach being oriented towards robustness to varying density, we evaluate our method's robustness to different point distributions. We test the methods (only at inference) with subsampled point clouds (randomly keeping one point per voxel of a 3D grid) and compare their segmentation performances (see Figure 8). Although the grid resolution varies, the number of points per sample block ($10m \times 10m$ column) stays the same. Since a fixed number of points per batch is needed (to make the algorithm parallelizable), point are repeated if necessary. Hence, increasing the sampling grid resolution also increases the redundancy of points in the input batches.

We notice in Figure 8 that our methods' segmentation performances are less impaired by large grid subsamplings than the other two methods. This effect is due to a combination of factors: our neighborhood selection is fixed (the neighborhood size is not dependent on point-to-point distances), we use a local spatial pooling before convolution (discarding redundant points). Interestingly, ConvPoint performs slightly better with a 5cm sampling, probably benefiting from the global density regularization induced by this type of sampling without too much a loss of information.

These results are promising for our approach especially considering the industrial quantities of data to be processed. To tackle this issue, the inference protocol could be redesigned to perform on roughly subsampled data (e.g. 30cm grid sampling) using a much larger block size to provide a fast global segmentation of very large point clouds with a minimal loss of performances e.g. for a fast count of catenary poles in a very large area.

6. CONCLUSION

In this paper we investigated the relevance of a deep-learning based approach for LIDAR railway PCSS with industrial standards. The railway environment is specific regarding the spatial disposition of infrastructure objects. We build on this specificity to design an axially symmetrical learnable kernel operator for integrating in a deep-learning framework.

We experimented with a classic U-Net architecture on two distinct LIDAR railway datasets. Our methods shows similar or better segmentation quality results (especially on railway specific objects of interest) compared to other state-of-the-art methods (cf. Section 5.2) and mitigated results in terms of convergence times on complex datasets (cf. Table 3). Finally, its ability to process strongly subsampled clouds without suffering too severe of a drop in segmentation performances opens industrial possibilities in terms of speed for specific maintenance use cases.

Future investigations into the network architecture could prove useful in scaling to industrial expectations as railway environments often resemble the Saint-Etienne in complexity. Several network architectures for PCSS show conclusive results as ResNets (Thomas et al., 2019) or attention mechanisms (Hu et al., 2020) and could support our kernel approach and speed up convergence on difficult datasets, hence bringing it closer to industrial standards. An other development to further evaluate our approach will consist in completing both datasets with additional data, compensating for the spatial and statistical class imbalances.

References

Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J., 2019. SemanticKITTI: A dataset for semantic scene understanding of lidar sequences. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Seoul, Korea, 9297–9307.

Boulch, A., 2020. ConvPoint: Continuous convolutions for point cloud processing. *Computers & Graphics*, 88, 24–34.

Boulch, A., Guerry, J., Le Saux, B., Audebert, N., 2018. SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks. *Computers & Graphics*, 71, 189–198.

Fey, M., 2020. PyTorch Cluster package (v1.5.9), *GitHub repository*. https://github.com/rusty1s/pytorch_cluster.

Guinard, S. A., Riant, J.-P., Michelin, J.-C., Costa D'Aguiar, S., 2021. Fast Weakly Supervised Detection of Railway-Related Infrastructures in LIDAR Acquisitions. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2021, 27–34.

Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A., 2020. RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Hua, B.-S., Tran, M.-K., Yeung, S.-K., 2018. Pointwise convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 984–993.

Kölle, M., Laupheimer, D., Schmohl, S., Haala, N., Rottensteiner, F., Wegner, J. D., Ledoux, H., 2021. The Hessigheim 3D (H3D) benchmark on semantic segmentation of high-resolution 3D point clouds and textured meshes from UAV LiDAR and Multi-View-Stereo. *ISPRS Open Journal of Photogrammetry and Remote Sensing*, 1, 11.

Lamas, D., Soilán, M., Grandío, J., Riveiro, B., 2021. Automatic Point Cloud Semantic Segmentation of Complex Railway Environments. *Remote Sensing*, 13(12), 2332.

Qi, C. R., Su, H., Mo, K., Guibas, L. J., 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Honolulu, HI, USA, 652–660.

Qi, C. R., Yi, L., Su, H., Guibas, L. J., 2017b. PointNet++: deep hierarchical feature learning on point sets in a metric space. *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, Curran Associates Inc., 5105–5114.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*, Springer, 234–241.

Soilán, M., Sánchez-Rodríguez, A., del Río-Barral, P., Perez-Collazo, C., Arias, P., Riveiro, B., 2019. Review of laser scanning technologies and their applications for road and railway infrastructure monitoring. *Infrastructures*, 4(4), 58.

Soilán, M., Nóvoa, A., Sánchez-Rodríguez, A., Riveiro, B., Arias, P., 2020. Semantic segmentation of point clouds with Pointnet and KPConv architectures applied to railway tunnels. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2020, 281–288.

Tchapmi, L., Choy, C., Armeni, I., Gwak, J., Savarese, S., 2017. Segcloud: Semantic segmentation of 3d point clouds. *2017 international conference on 3D vision (3DV)*, IEEE, 537–547.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., Guibas, L. J., 2019. Kpconv: Flexible and deformable convolution for point clouds. *Proceedings of the IEEE/CVF international conference on computer vision*, 6411–6420.

Zhang, Z., Hua, B.-S., Yeung, S.-K., 2019. ShellNet: Efficient Point Cloud Convolutional Neural Networks Using Concentric Shells Statistics. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Seoul, Korea, 1607–1616.

Zhu, X., Zhou, H., Wang, T., Hong, F., Ma, Y., Li, W., Li, H., Lin, D., 2021. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9939–9948.