Semantic segmentation of industrial facility point cloud
-
EDF Challenge

Loick Chambon loick.chambon@eleves.encp.fr

September 28, 2022

Contents

1 Introduction:

The goal of this challenge is to perform a semantic segmentation of a 3D point cloud. It consists of giving a label to each point of a scene. It is the same task as object part segmentation but on real scenes instead of single objects. It is thus more suited for real applications. 3D semantic segmentation is a well-known problem useful for urban scene modeling, interior scanning, urban planning and industrial modeling and several datasets exist (Tab. 1). There are challenging datasets with many acquisition means and environments.

Name	*Environment*	*Acquisition*	*Colors*	*Training*	*Test*	*Classes*
Paris-Lille 3D [6]	Outdoor	Mobile lidar	No	140M	30M	50
Semantic 3D [4]	Outdoor	Fixed lidar	Yes	1.660M	2.349M	8
S3DIS [1]	Indoor	Depth cameras	Yes	273M	-	13
Scannet [2]	Indoor	Depth cameras	Yes	5.521M	449M	20

Table 1: 3D Semantic Segmentation Datasets

According to this taxonomy, EDF has provided a dataset acquired in an industrial environment, the boiling room of EDF Lab Saclay whose design is sufficiently close to an industrial building. The boiling room was digitized with LiDAR scanners on tripods and contains 68 scanner stations. Each acquisition at one scanner position produces one point cloud of about 30 millions of points. The provided dataset contains 170M of points whose 130M belong to the training set. The points are labelled with 10 different classes: *"Background"*, *"Beams"*, *"Cabletrays"*, *"Civils"*, *"Gratings"*, *"Guardrails"*, *"Hvac"*, *"Ladders"*, *"Pipping"*, *"Supports"*. A detailed analysis of the data will be performed later with a particular look at the distribution of classes.

Mathematically, the task is to solve the following problem: find a function g such that for every point x_i in point clouds of size N, $\mathcal{P} = \{(x_i, f_i) \in \mathbb{R}^3 \times \mathbb{R}^D\}_{i<N}$ where D represents the number of additional features of each points, we can retrieve its label y_i representing one class among 10:

$$\forall (x_i, f_i) \in \mathbb{R}^3 \times \mathbb{R}^D, g(x_i, f_i) = y_i \in \{0, ..., 9\}$$

The additional features may contains RGB colors, intensity or hand-crafted features.

2 Exploratory data analysis

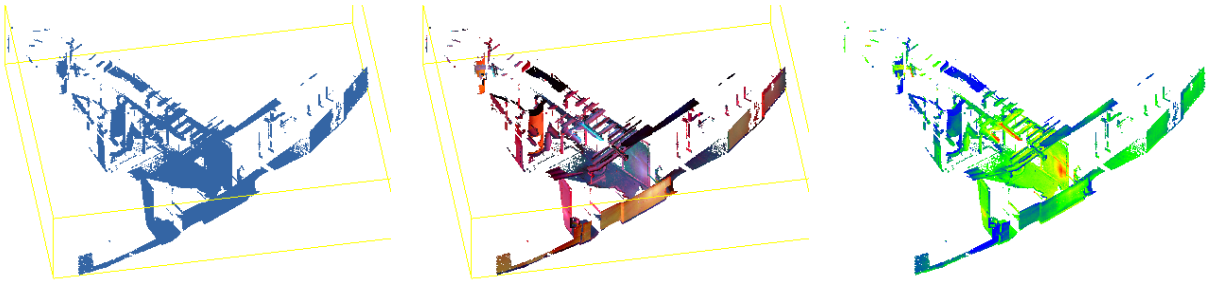
Before to train a model, we have explored the dataset in order to find its specificities. It contains 68 scans splitted in two sets, a training set with 50 scans and a testing set with the remaining scans. In each scan, the points are represented with seven features:

- a triplet of 3D spatial coordinates in a global reference frame having a long range (Tab. 2);
- a scalar value corresponding to the intensity return of the laser beam;
- RGB values corresponding to the reflected color.

Visually, it is not clear that color and intensity leads to relevant information (Fig. 1). Points are only colored in blue, red and orange and colors are not consistent. Some pipes have blue and red with a kind of color gradient. Hence it is hard to discriminate classes using only colors. It is worse

Features	x	y	z	r	g	b	i
Min	1.97	78.96	146136.00	0	0	0	0
Max	36319.70	17475.70	155130.00	255	255	255	255
Mean	7597.20	9669.76	1.52	134.74	113.14	113.76	112.18

Table 2: Range of features calculated over all available scans



(a) Raw point cloud (b) RGB point cloud (c) Intensity point cloud

Figure 1: Three different points of view of the 10th scan.

with intensity because the feature is link with the acquisition and not with the labels. Nonetheless, it can be helpful to determine local densities and a neural network can find a better interpretation.

About statistics, we have performed several analysis to study characteristics of the point clouds. We have seen that:

- Scans do not contain the same number of points. In average there are 2.6M points in a scan. The values go from 1.6M to 4.1M. The differences come from the parameters of the acquisition. The scanner may not have been in operation during the same amount of time or with the exact same configuration between different scans. Moreover, some points in scans are far away from others. They are isolated and can be seen as outliers if we do not consider the spatial correlation between the scans. The presence of such points can complicate our models since a local analysis cannot be performed due to isolation and a global analysis requires a huge receptive field that may not fit in memory;
- Labels are not equally distributed among scans (Fig. 2). The most represented classes are "Civils" (44%), "Pipping" (30%) and "Background" (9%) and the under-represented classes are "Ladders" (1%), "Beams" (1 %) and "Guardrails" (1.4 %). It explains why the benchmark takes the most represented class, assigns each point to that class and obtain 34% of weighted F1-score;

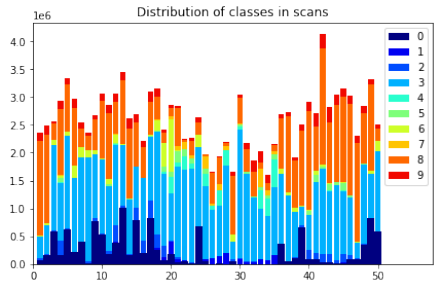
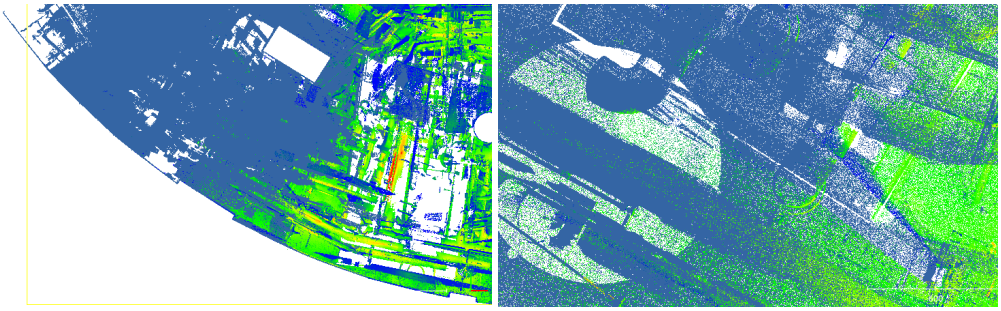


Figure 2: Representativeness of the classes according to the scans.

- Scans are spatially entangled and the test scans are also entangled with the traning scans. It is a strange characteristic of our dataset but it implies that we have some local information available for our testing set. Moreover, the test set is the extension of the training set which implies a border (Fig. 3). We do the choice to not use this information to learn our mapping function because it implies to have additional information from the testing set and in a canonical problem we do not have it;



(a) Border (b) Zoom in a local test region.

Figure 3: Entanglement of the training and the testing sets. Test points are in blue and training points are colored with their intensities.

3 Experiments

3.1 General overview

Computer vision algorithms use representations to create an underlying model of the reality. The representations can be more or less elaborate making the underlying model more or less abstract. When looking at computer vision algorithms, with this perspective, we can classify each 3D semantic segmentation algorithm according to the degree of abstraction of the model:

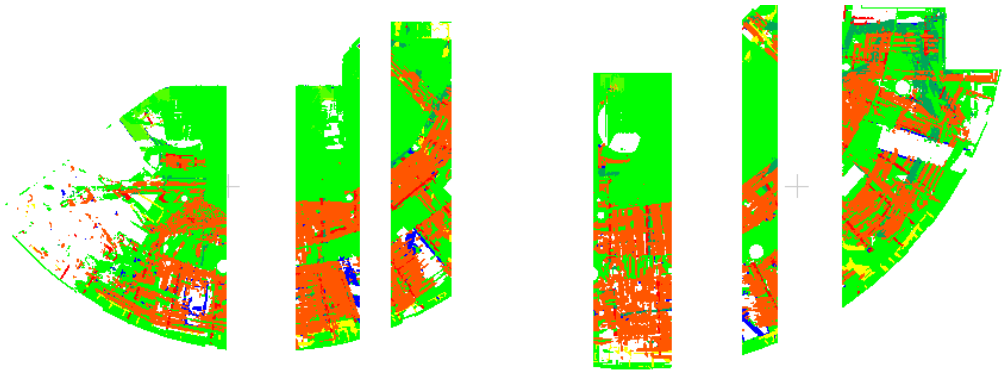
- Modelling objects: the objects are detected as simple primitives like planes, spheres, cylinders, cones and tori. We can cite RANSAC [3] which is a voting method on random surface samples using a quorum of points to represent the primitive, or region growing algorithms which uses local proximity of points and similarity criteria;
- Modelling geometric patterns: point clouds are described by hand-crafted features. The underlying model is built using a machine learning algorithm that determines which values of these features correspond to which semantic classes. Hand-crafted features such as verticality, linearity, planarity and sphericity, can be computed using the eigenvalues of the covariance matrix of local point clouds. For example, a modern approach [8] uses a random forest with a multiscale spherical neighborhoods to perform semantic segmentation on an outdoor dataset;
- Modelling learnable representations: the point clouds are processed by a neural network that learns its own features in order to find the right mapping function from the point cloud to the labels.

After some preliminary experiments, we have concluded that RANSAC is not suited for complex shapes so we have discarded this approach. And, after a review of the litterature, we have decided to use a deep learning network called *KPConv* [7].

3.2 Pre-processing

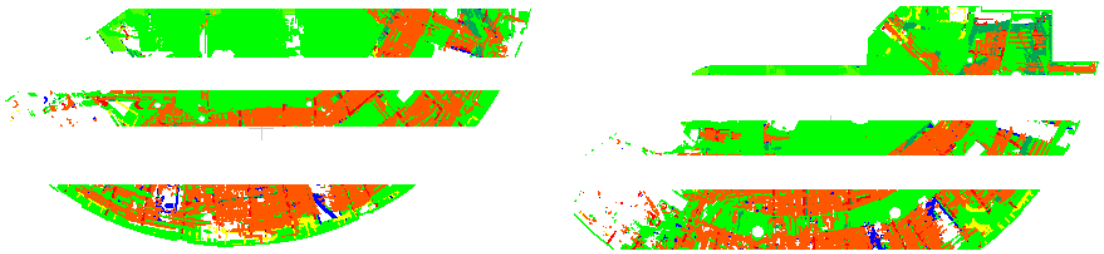
Before processing our point clouds with the network and explaining how it works, we preprocessed the data to make it more canonical.
First, we gathered the point clouds, sorted them according to their coordinates before to split them. The goal is to avoid isolated points and to avoid overlapping training scans. With this setting, training scans can be considered as independent scenes separated by borders.

- Splits: we have tested two different configurations depending on the coordinates on which we have sorted the point cloud. The first is a split on x (Fig. 4), the second on y (Fig. 5). We decided not to separate the points according to the z-coordinate because the borders would share a larger area. Another viable strategy would have been to separate the scans according to an angular angle as the data form an arc of a circle but the estimation of the center of the circle is somewhat uncertain. Moreover, this strategy is really dependent of our dataset and can not be generalized.
- Number of splits: since we have decided to use a neural network, we studied what was a configuration that worked with it and we focused on the S3DIS example for 3D semantic segmentation [1]. After preprocessing, S3DIS contains 273M points in 6 areas with about 45M points per area, one area is used as a validation area. Thus, for the x-split and the y-split, we have decided to divide our dataset into 6 training scans (5 purely training and 1 for validation) containing about 21M points each in order to have an equal number of points between scans. 21M is less than 45M but it is the same order of magnitude, so we did not look for another split. The test scans were also gathered and split into 2 scans containing about 20M points. Since the coordinates are not altered we are sure to be able to trace back the points.



(a) Scans "0", "2" and "4" of the x-split (b) Scans "1", "3" and "5" of the x-split

Figure 4: Training scans of the x-split. They have the same number of points but does not cover the same area.



(a) Scans "0", "2" and "4" of the y-split (b) Scans "1", "3" and "5" of the y-split

Figure 5: Training scans of the y-split. The scans have a larger border area compared to the x-split.

In addition, we have performed data augmentation on the fly. Since our dataset is distributed in an arc of a circle, Z-rotations, and XY-flipping are usefull. Noise is almost mandatory to make sure we do not overfit. Scaling is also used because we have considered that objects belonging to the same class have varying size.

3.3 Processing

The chosen network was introduced by H.Thomas during a thesis at Mines de Paris two years ago in the paper *KPConv: Flexible and Deformable Convolution for Point Clouds* [7].

3.3.1 Model

Intuition: The idea behind KPConv is similar to the one behind CNNs. In CNN, each pixel is represented by a list of channels and is passed through k filters. The pixel's new representation is the dot product of neighboring pixel's channels by the filters. Similarly, in a KPConv layer, each point has features and is multiplied by k kernel points (similar to filters). The new representation of the point is the sum of all the kernel values multiplied by the neighbor's features. The method can not be easily generalized since there are intrinsic differences between points and pixels. Especially, point clouds are disordered, sparse and contain more data. However, there are several tips for generalizing:

- Unlike the images, since the points are unordered, a neighboring point in the list is not a neighboring point in space. To obtain neighborhoods, the authors chose to use a KDTree with a ball radius query.
- The sparsity of the data implies that the number of points in a local neighborhoods vary and that it is linked with the density of the data. To control the density of input points at each layer of the network they use a grid subsampling strategy with a cell size dl_0 while building a KD-Tree in order to trace back the points. Thus, the support points of each layer, carrying the features, are chosen as barycenters of the original input points contained in all non-empty grid cells (Fig. 6). Another strategy is to take the center of the grid cell instead of the barycenter.
  The initial cell size is an important parameter of the model. To set it, we have looked at the value of the cell size for the S3DIS dataset, it is 0.03m. Then we have implemented a grid

subsampling to get the ratio of not empty voxels in the grid. It corresponds to 90%. That means, with this typical distance, 90% of the points are in a non-empty grid cell. Finally, we have looked for the value that leads to 90% of filling in our dataset. It leads us to 4m. Finaly, we chose a slightly lower number 2.8m to have even more context.

- To control the number of points, they have chosen to reduce the number of points progressively. To do so, they double the cell size $dl_{i+1} = 2.dl_i$ of the grid subsampling at every pooling layer, along with the other related parameters, incrementally increasing the receptive field of KPConv layer. In the literature, other subsampling methods exist. We can cite *farthest point sampling* used in PointNet++ [5] that samples points iteratively in a point cloud using a distance critera. This strategy helps picking points regularly across the whole point cloud and ensures a minimal distance between them. However, it has a complexity of $O(n^2)$

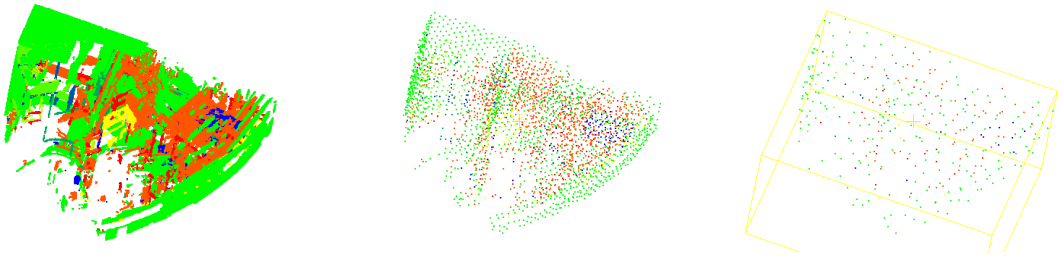


Figure 6: Three different grid subsampling of a scan with varying minimal space between points using the CloudCompare tool.

After preprocessing, the points are processed by the network (Fig. 9) that introduces a new layer: the KPConv layer. We present the main idea of the network in order to understand the different steps, but more details can be found in the original paper and in the thesis of the author.

Ideas: The KPConv layer is a kind of convolutional layer for 3D point clouds. It shares many ideas with image convolutions and the formulation of a point convolution is inspired from images convolutions while having major differences because a point is sparse and continuous, while an image is dense and discrete. If we call x_i the points from $\mathcal{P} \in \mathbb{R}^{N\times 3}$ and f_i their corresponding features from $\mathcal{F} \in \mathbb{R}^{N\times D}$, then, the general point convolution of $\mathcal{F}$ by a kernel g at a point $x \in \mathbb{R}^3$ is defined as:

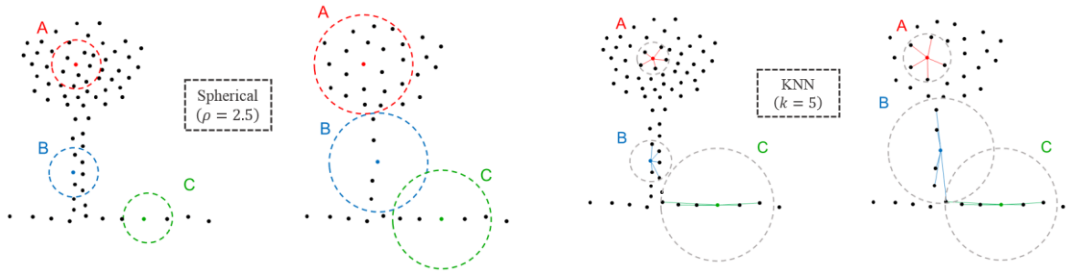
$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} g(x_i - x) f_i$$

Where the neighborhoods are defined by local sphere parametrised by r and centered on the query point:

$$\mathcal{N}_x = \{x_i \in \mathcal{P}, ||x_i - x|| \leq r\}$$

The typical radius is a key parameter of the layer. By default, it is chosen to be $r_j = \rho \times dl_j$ where dl_j has been previously introduced as the grid sub-sampling cell size and ρ is an hyper-parameter of the model sets to 5 as a tradeoff between performance and computation speed. For 3D semantic segmentation, a rule of thumb states that $r = 50 \times dl_0$ is a suitable parameter. If we increase the coefficient, we can get more local information. Considering the size of our objects and the available memory of our GPU, we have chosen r to be equal to $75 \times dl_0$.
About the neighborhoods, another option would have been to use a KNN. It has the advantage to

give a fix number of point in the network and it ensures a steady computing speed and fixed matrices sizes for GPUs. However it does not take into account the local density and the sparsity of the data (Fig. 7). Moreover, it can assign a point far from the center only because we need K points.



(a) Spherical neighborhoods through scales

(b) KNN neighborhoods through scales

Figure 7: KNN leads to an increasing receptive field and can match points that are far from each other while spherical neighborhoods have a variable number of points but have always the same size of receptive field.

Before to apply the kernel, we expressed our 3D points in a local referential, according to the center of the sphere and call the new coordinates by $y_i = x_i - x$. Then, we build the kernel such that we apply different weights to different areas inside the local ball $\mathcal{B}_r^3$. The authors have chosen to define their kernels using K kernel points: $\{\hat{x}_k | k \leq K\} \in \mathcal{B}_r^3$ where K is an hyperparameter of the model and equals 15. The paper indicates that increasing this value brings little benefit since it is the plateau value at which the evaluation metric stagnates. Then, by defining the associated weights matrices that map features from dimension D_{in} to D_{out} we can write:

$$g(y_i) = \sum_{k \leq K} h(y_i; \hat{x}_k) W_k$$

where h expressed the correlation between $\hat{x}_k$ and y_i. It has to be high when the points are closed. In order to facilitate the gradient back-propagation, the authors have chosen to set h by a linear correlation instead of a gaussian correlation or a rectangular correlation:

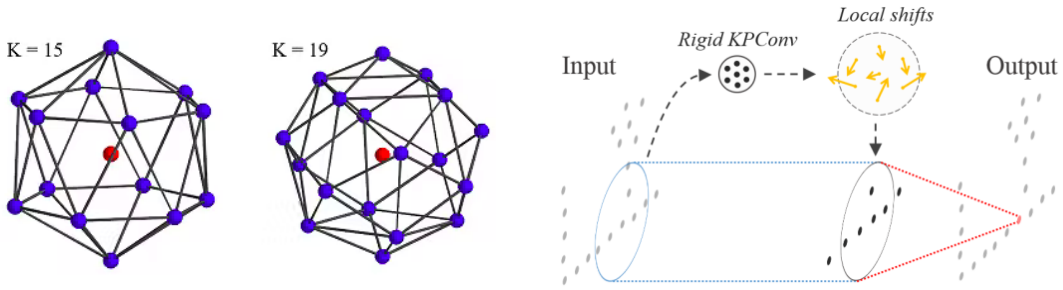
$$h(y_i; \hat{x}_k) = \max(0, 1 - \frac{||y_i - \hat{x}_k||}{\sigma})$$

where σ represents the influence distance of the kernel points, and is chosen according to the input density to be equal to: $\sigma_j = \Sigma \times dl_j$. Σ is an hyperparameter of the model equals to 1.0 in the paper. It controls the density of points that our convolution can see because when it increases, h is positive for more points and is therefore taken into consideration in the calculation of the sum. About the kernel points $\hat{x}_k$, their positions are either rigid or deformable and leads to two different models. During our challenge, we have used the deformable model that leads to better results in the S3DIS benchmark.

- The rigid model positions the kernel points to cover the largest possible area. To do so, they use a "cristal" structure where each point applies a repulsive force on the others.
- The robust model applies a rigid KPConv layer that maps D_{in} features to $3K$ values in order to output the local shifts $\Delta(x)$ (Fig. 8). During training, the network learns the rigid kernel

generating the shifts and the deformable kernel generating the output features simultaneously where the new kernel formula is given by:

$$g_{deform}(y_i, \Delta(x)) = \sum_{k<K} h(y_i; \hat{x}_k + \Delta_k(x))W_k$$



(a) Kernel points positions in a rigid structure

(b) Deformable operations

Figure 8: Rigid KPConv. The deformable layer applies a rigid KPConv to get the shifts before to use them to compute the kernel that is applied on the input

Finally, given our KPConv layer, we have used the KP-FCNN architecture (Fig. 9) of the original paper for semantic segmentation. Considering what has been said, it is used to segment small subclouds contained in spheres. At training, the spheres are picked regularly in the scenes in order to be robust to varying densities. The pickled centers are called potentials. At testing, we perform a voting strategy by picking spheres regularly in the point clouds and we stop the process when every points are tested multiple times by different sphere locations. Then we average the predicted probabilities for each point.

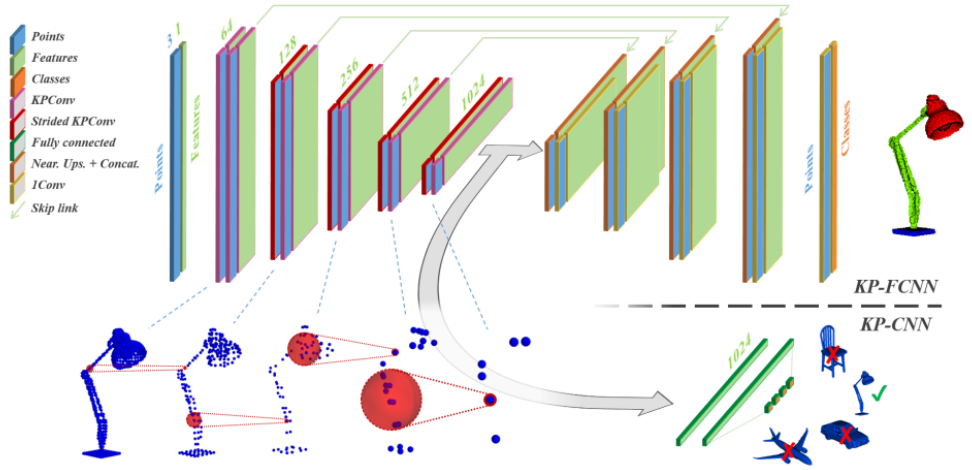


Figure 9: Architecture of the model. The originality of the paper is the introduction of the KPConv layer and its strided version. There are two different architectures depending on the task: KP-FCNN for segmentation and KP-CNN for classification.

3.3.2 Implementation

I have spend many hours to be able to setup the KPConv network with the EDF dataset using torchpoints_3d [1]. Torchpoints 3D appeared as a good starting point because it allows multiple data-transformations and have implemented several networks and modules. I was able to modify classes for dataloading, transformations, pre-filtering etc. However, when I finished the preprocessing, the pipeline was not completely satisfactory and I did not have enough control over the network to be able to modify parameters and to understand the core of the network. I lost a lot of hours but learned a lot about building a python library and about how to process point clouds in Torch. Then, I have decided to use the official Pytorch implementation [2]. It was far more easy to setup and very documented with the help of the github issues (no official documentation exists). Since I don't have a GPU, I used Google Colab Pro P100 with high-RAM for my experiments. Each train takes 10 hours to complete, so my experiments were limited by runtime. However, with patience, I was able to find a good setting that leads to promising results. My experiments lasted 150-200 training epochs even though the implementation advises to use 500 epochs. In practice, if I had used the session longer, I would have risked getting kicked. About testing, it takes me 3 hours to get the final vote predictions and I used 20 votes instead of the default 100 to speed up the testing phase.

3.4 Evaluation

We have evaluated our network using a weighted F1-score as mentioned in the challenge proposal:

$$F_1 = \sum_{i=0}^{C-1} w_i \frac{P_i \times R_i}{P_i + R_i}$$

where P represents the precision, R the recall and w_i is the number of true instances for each label divided by the number of points. Weights are important since we have an unbalanced dataset and are handle with a scikit-learn parameter of the f1-score metric. Moreover, since in the litterature, IoU is prefered, we have also evaluated our predictions using IoU.

$$IoU = \frac{F_1}{2 - F_1}$$

Finally, to see where our errors lie and whether the training phase was actually learning, we plotted the confusion matrix.

4 Results

With our method, we obtained with x-split a public score of **0.9401 which ranks 1st** on the public leaderboard and a private score of **0.9400 which ranks 1st** on the private leaderboard. To analyse our errors and our predictions, we use CloudCompare visualisations and compute some metrics on the validation set composed of one gathered point clouds that contains 21M of points. To chose the validation set, we have calculated what is the percentage of classes in each scans. Since scans have the same number of points, they densities are comparable.

The classes are not equally distributed. This is logical since the scans are taken in an industrial environment. In such an environment, objects are not randomly positioned and follow a spatial

[1] https://torch-points3d.readthedocs.io/en/latest/src/tutorials.html
[2] https://github.com/HuguesTHOMAS/KPConv-PyTorch

Scans	Background	Beams	Cabletrays	Civils	Gratings	Guardrails	Hvac	Ladders	Pipping	Supports
0	6.56	0.46	1.74	42.71	0.89	1.41	1.48	0.0	39.52	5.21
1	0.98	1.79	1.29	41.39	5.25	2.32	0.54	3.59	37.06	5.79
2	7.9	1.11	1.65	46.26	2.82	1.9	1.69	1.05	31.17	4.45
3	8.84	2.29	2.34	41.53	4.03	2.33	6.76	1.03	29.03	1.83
4	13.8	0.46	3.58	50.93	0.16	0.45	5.1	0.33	20.78	4.41
5	16.59	0	4.38	46.89	0	0.01	1.76	0	25.42	4.94

Table 3: Percentage of classes in the different scans of the training dataset. One validation scan is chosen.

structure and coherence. Since our scans are spatially distributed, having so many classes per scan would mean having objects equally distributed across the site, which is not the case. To chose our validation scans, we have eliminated the scans 0 and 5 because they do not contain every classes and we took depending on the configurations, one scan among the remaining (in general the 5th). Then, during training, we have saved the evaluated predictions every 50 epochs in order to follow the evolution of the accuracy of the predictions.

4.1 Confusion matrix

The confusion matrices help us to understand what are confusing classes for the network (Fig. 10, 11).

- After the 50th epochs it did not learn to classify beams, hvac and ladders. Each are underrepresented classes of the validation scan (5th scan), and counts for respectively: 0.46%, 5.1% and 0.33% of the classes. Furthermore, after 50 epochs, the network fails to distinguish background and hvac and confuses guardrails with ladders. The network was not trained enough and yields poor results.
- After the 150th epochs, it impressively learnt to classify most of the classes. It has some difficulties to distinguish guardrails with beams and supports with background. Apart from that, the results are really impressive.

4.2 Metrics

We have evaluated our validation scan using weighted F1-score of scikit-learn and we have obtained 0.86 (Fig. 4). It is lower than the score obtained in the public leaderboard equals to 0.94. About IoU we reached 0.76 and comparing to our training IoU, we have overfitted a little.

Epochs	50	100	150
F1	0.673	0.743	0.86
IoU	0.507	0.591	0.753

Table 4: Metrics for our validation set

We have also evaluated our y-split. It returns worse results on the public F1-score). We believe that this is due to the fact that the area of the borders between the scans is larger than in the x-split (Fig. 5), and these borders correspond to areas of high uncertainty because points have less contextual information than when they are surrounded by other points. The F1-score and the IoU are 0.60 and 0.75 on the validation set and we obtained a score of 0.896 on the public leaderboard (which is still better than the second).

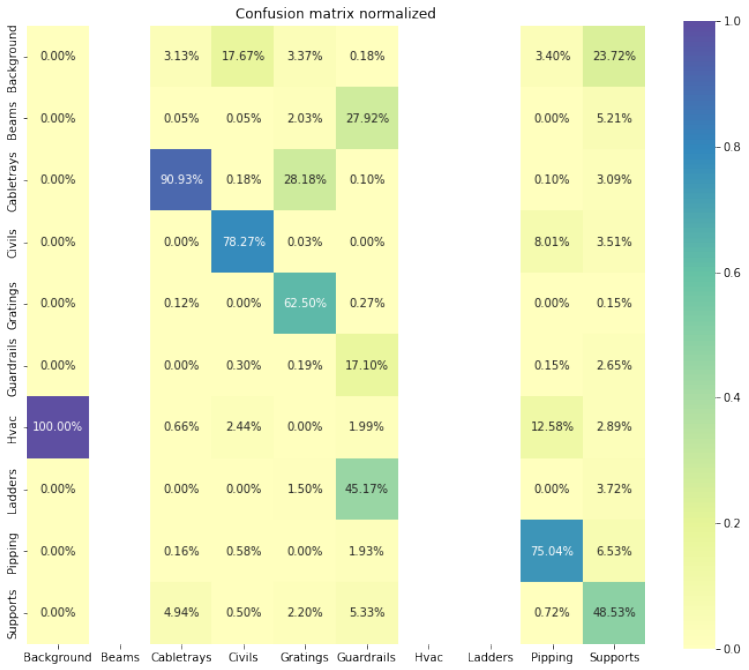


Figure 10: Confusion matrix normalized by columns, after the 50th epochs of the training process.

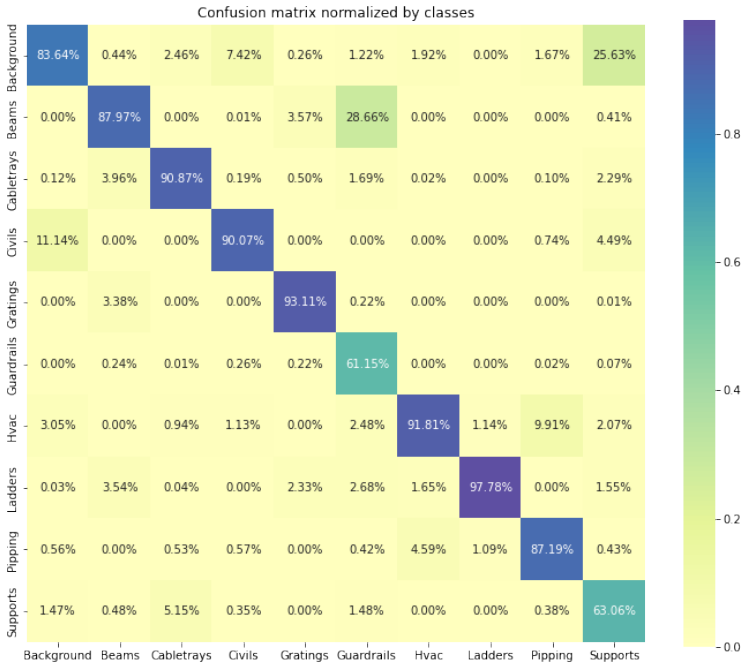
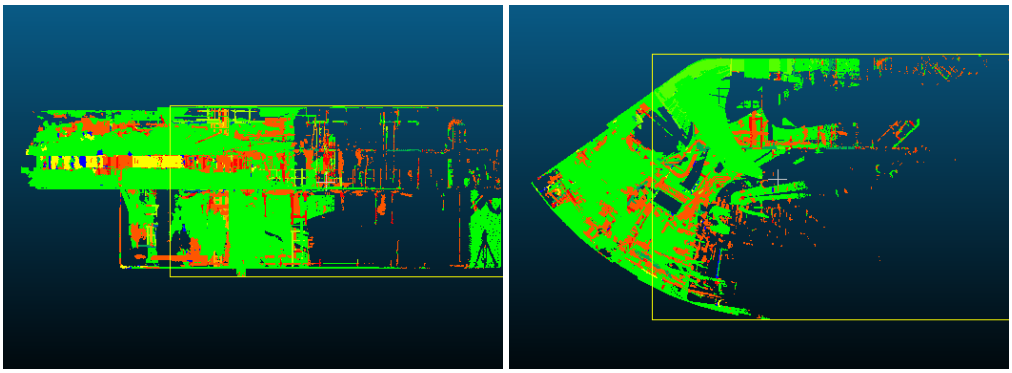


Figure 11: Confusion matrix normalized by columns, after the 150th epoch of the training process.

4.3 Visualisation

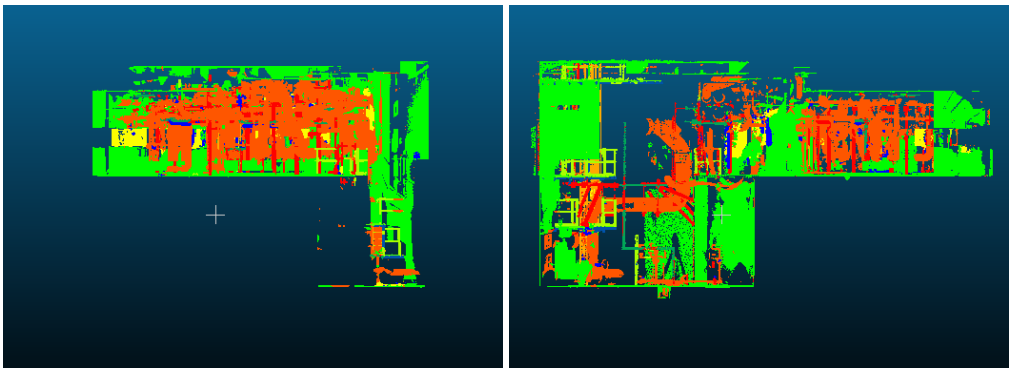
Our test set consists of two scans as explained in the pre-processing sub-section. Since our two scans share a border, it is interesting to see if the network predicts the same labels for both parts of the boundary (Fig. 13). It is globally the case but we have different colors in some places. To avoid

these artefacts, it should have been better to use overlapping test and to label the points with the most probable class either the maximum of the values or the maximum of the sum of probabilities. Nonetheless, it is only a refinement of the method and only improves the local rendering of the border.



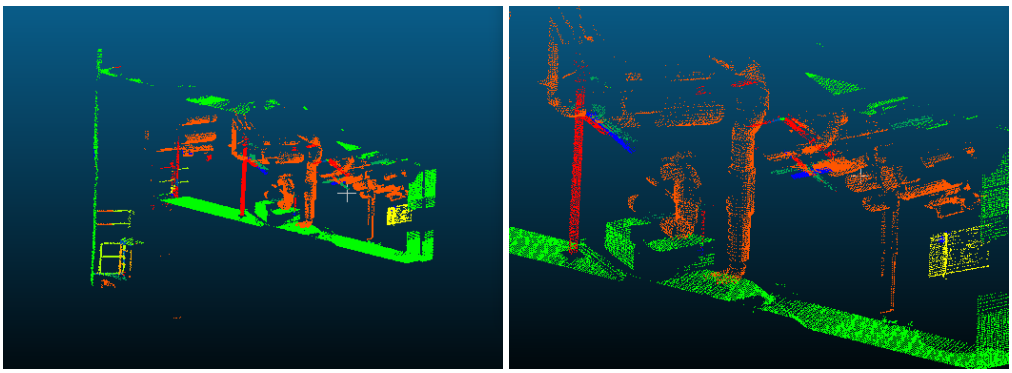
(a) Front view (b) Top view

Figure 12: Different point of view of the final test point cloud.



(a) Left side of the boundary (b) Right side of the boundary

Figure 13: Two different views of the boundary.



(a) Boundary between the two scans (b) Zoom to exhibit local differences

Figure 14: Section of the boundary between the two scans. It is not perfect. We can see that points labelled in red from one side are labelled in orange from the other.

5 Conclusion

The challenge was not easy because I have used a difficult and recent method with limited resources. KPConv yields impressive results on the public leaderboard and allowed me to rank 1st on the private leaderboard but requires several hours to be trained. Once the model is trained, inference is not automatic since it is based on a voting scheme using potentials points. The network is not suited for real time inference. However, two years after its release, it is still state of the art and shows impressive results on huge benchmark datasets for different environments: indoor and outdoor. Moreover, it is very easy, once you have understood how it works, to train and evaluate the network on a custom dataset, even if it has a very different density compared with the benchmark datasets.
During this challenge, I decided not to use the dataset bias of the entanglement between training and test scans because I have assumed that in real-world applications, it is really rare to need to segment an incomplete, labeled point cloud. I suppose if we had extended the local information to the surrounding points, we could have obtained more consistent results.
With more time and resources, I could have tried to go deeper into finding hyper-parameters and studying the influence of color and intensity.

References

[1] I. Armeni et al. *Joint 2D-3D-Semantic Data for Indoor Scene Understanding*. Feb. 2017. arXiv: 1702.01105 [cs.CV].

[2] Angela Dai et al. *ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes*. 2017. arXiv: 1702.04405 [cs.CV].

[3] Martin A. Fischler and Robert C. Bolles. *Random Sample Consensus: A Paradigm for Model Fitting with Apphcatlons to Image Analysis and Automated Cartography*. 1981.

[4] Timo Hackel et al. "SEMANTIC3D.NET: A new large-scale point cloud classification benchmark". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. IV-1-W1. 2017, pp. 91–98.

[5] Charles R Qi et al. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *arXiv preprint arXiv:1706.02413* (2017).

[6] Xavier Roynard, Jean-Emmanuel Deschaud, and François Goulette. "Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification". In: *The International Journal of Robotics Research* 37.6 (2018), pp. 545–557. DOI: 10.1177/0278364918767506.

[7] Hugues Thomas et al. "KPConv: Flexible and Deformable Convolution for Point Clouds". In: *Proceedings of the IEEE International Conference on Computer Vision* (2019).

[8] Hugues Thomas et al. "Semantic Classification of 3D Point Clouds with Multiscale Spherical Neighborhoods". In: (2018).