RangeNet++ from LiDAR scans to outdoor point clouds

Théo VINCENT
ENS Paris Saclay & ENPC
theo.vincent@eleves.enpc.fr

Abstract

This report brings two major contributions. First, it summarizes the algorithm RangeNet++ in a concise way. Then, it introduces a new method for segmenting a 3D outdoor point cloud using the RangeNet++ algorithm. This method allows the user to generate an infinite number of virtual scans on which the RangeNet++ network can train on. The overall algorithm archives satisfying results on the benchmark of 3D Point Cloud classification for Master NPM3D Course https://npm3d.fr/benchmark-for-master-course-on-3d-point-clouds.
The code is available at https://github.com/theovincent/3DPointCloudClassification.

1 The algorithm: RangeNet++

We are going to go through the algorithm RangeNet++. It aims at labelling a LiDAR scan. Interested readers can have a look to the paper of A. Milioto et al. [4] for additionnal details. The algorithm can be split into four parts, this is why this section has four subsections. This summary of the algorithm comes with a fork of the original code https://github.com/theovincent/lidar-bonnetal that allows the user the run the code without bugs.

1.1 Point cloud to range image

The main idea of RangeNet++ is to use the fact that LiDAR scans are acquired from only one point in space so the information can be represented on a 2D space. This is the reason why the first step of the algorithm is to apply a spherical projection on the 3D points. Thanks to this transformation, they can leverage the advances made in the field of deep learning for images. In order not to lose information, for each scan, they make five 2D arrays that are representing the x, y, z coordinates, the distance to the camera which is called the range and the remission. The remission is the intensity of the returning beam that the Velodyne sensors. These five 2D arrays can be represented by a tensor that they call range image this is why this algorithm is called RangeNet.

The dataset used in the paper is the SemanticKITTI dataset [2]. It is composed of 43 000 scans and 28 classes were labelled. To craft it, they use Velodyne sensors of 64 beams recording around 120 000 points per scan. These technical details allow us to understand the shape of the images that are used. Indeed, the images are of size [64 x W] where W is the width of the image that can be chosen freely. The smaller it is, the smaller the neural network will be, but also the bigger the loss of information we will be. The loss of information is due to the fact that the spherical projection brings the 3D points recorded by one beam on a mesh of W boxes. This means that if W is too low, several points will be in the same box. In this case, there will be a loss of information. The way this problem is handled is dealt in subsection 1.4. To know which point to assign to the pixel, they choose the one with the smallest range. This assignment procedure is kept into memory to be used in subsection 1.3.

1.2 Semantic segmentation

As we start explaining it in the previous subsection, the authors use a neural network to predict the label of each pixel of range images. The output of the network is, for each pixel, the probability of being in a class, this forces the output shape to be [C x W x 64] where C is the number of classes (28 in the case of the SemanticKITTI dataset).

The chosen architecture is close to a U-Net [5] except that the max pooling are replaced by the down-sampling only over the width dimension since the height dimension is already small enough with 64 pixels. The loss used is a weighted cross entropy. The weights are defined for each class c by $w_c = \frac{1}{\log f_c}$, where f_c is the frequency of appearance of each class. This is to cope with the fact that the data is not well-balanced.

1.3 Range image to point cloud

Now that the range image has been passing through the network, the idea is to project back the information from the range image to the 3D points. To do so, the assignment procedure of the subsection 1.1 has been stored into memory so that a label can be assigned to every 3D point.

A problem appears when several 3D points had been assigned to one pixel. This phenomenon is problematic when the points assigned to the same pixel are coming from two different classes. In that case, it is as if the class would leak to the furthest points. This issue is solve by the following subsection.

1.4 Point cloud post-processing

The authors implemented a fast and light weight K-Nearest-Neighbors (KNN) search over the range image prediction. The idea is to be able to assign different labels to the points that have fallen into the same pixel. This KNN method takes into account the range and is using a Gaussian kernel with a threshold over the neighbors.

We then have an algorithm that takes as input a LiDAR scan and outputs for each point a prediction of a class.

2 From Outdoor Point Cloud to LiDAR scans

The main idea of this report is to show that the algorithm RangeNet++ previously, presented in section 1, can help us segment an outdoor point cloud. Let's consider that we have an outdoor point cloud. The way we can do this is by transforming this point cloud into many virtual scans. It is as if there would have been plenty of virtual Velodynes to would have sensored parts of the point cloud. Then, using the RangeNet++ algorithm, we can get the predictions for each 3D point of each virtual scan and merge them together to get a full segmentation of the initial point cloud. Figure 1 shows the end to end pipeline. I was inspired by the work of Y. Ali et al. [1] where they add an additionnal projection to the RangeNet++ algorithm to get better results.

We still have to see how we can generate a virtual LiDAR scan for the outdoor point cloud. These virtual scans need to have the same properties as the real LiDAR scans collected in the SemanticKITTI dataset. This step is necessary to be able to reuse pretrained weights coming from the paper of A. Milioto et al. [4]. This steps involves computing some characteristics of the SemanticKITTI dataset see subsection 2.1. Then, the same characteristics have to be computed for the outdoor point cloud, see subsection 2.2. Finally, the characteristics of the outdoor point cloud will be moved to the characteristics of the SemanticKITTI dataset by some transformations, see subsection 2.3.

In a more practical way, the goal of this step is to change the outdoor point cloud so that it looks similar to a LiDAR scan. To do so, the transformations that will be introduced act as if we would put a Velodyne inside the point cloud and see which points are reachable from there.

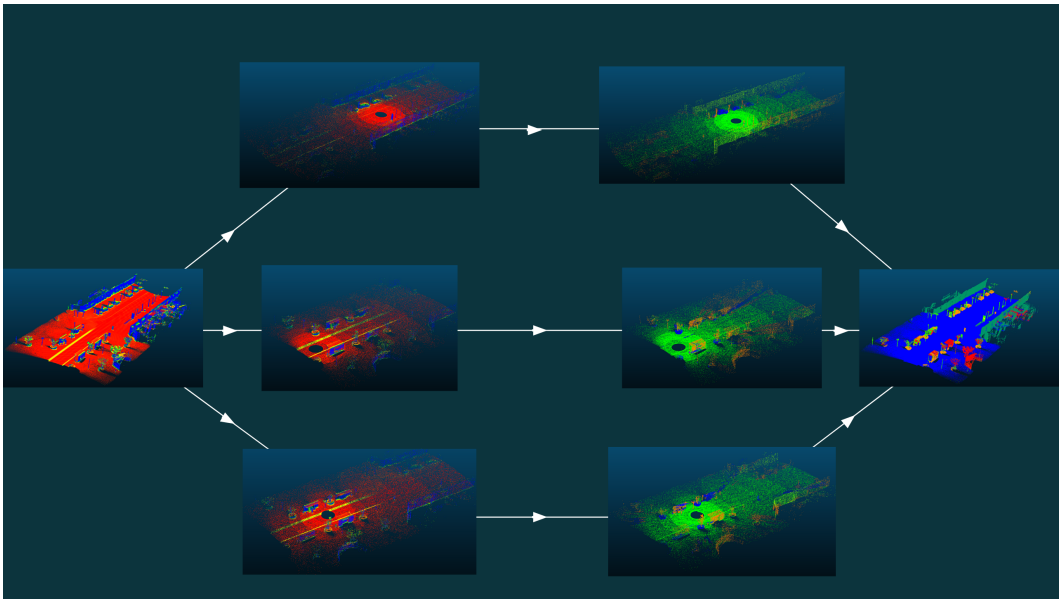


Figure 1: End to end pipeline. Starting from the raw point cloud with the feature verticality on the far left, the point cloud is split in several virtual scans (three here). Then those virtual scans are passed through the RangeNet++ algorithm to be labeled (middle right). The predictions are then merged together to have one prediction per 3D point. Here the final prediction in the far right is shown with 1000 merged virtual scans. An animation is also available on the GitHub repository.

2.1 SemanticKITTI characteristics

The SemanticKITTI dataset was recorded with a Velodyne on top of a car. This means that the ground is at a stable level on every scan. To compute this level, I chose to split all the points of each scans into bins. The bin with the most point would be the ground level. Since this value slightly changes throughout the dataset, I took the median over the values.

The same kind of reasoning was applied to compute the maximum and the minimum distance from the sensor and the maximum height of the points.

Another important characteristic to take into account is the number of points of the scans and more specifically the distribution of the points in the space. Indeed, as the Velodyne sends beams of light the get the information about the environment, the distribution of the points in the space is far from being uniform. For instance, more points will be sensored closer to the sensor. This is why I chose to model the distribution of the points as being defined by the concentric spheres of radius exponentially increasing with center the position of the virtual Velodyne. The compute the actual distribution, I went through all the scans to count the median number of points for each concentric sphere. This gives us a way of sampling the 3D points of the outdoor point cloud. Figure 2 shows how the space is split.

2.2 Outdoor point cloud characteristic

We first need to see where to place the virtual Velodyne. For that, we would need to have a rectangle of admissible centers. This rectangle lies only on two dimensions, x and y, since the height of the point is fixed by the height of the ground previously computed. Here is the only time of the overall process where human annotation is needed. The user needs to give the four vertexes of a rectangle where the virtual Velodynes can be placed. This procedure could be easily replaced by an estimation of the position of the road but I considered that the purpose of the project was not there. Note that it is important that the admissible centers are at a position where the real Velodyne of the SemanticKITTI dataset could have been. This means on the road.

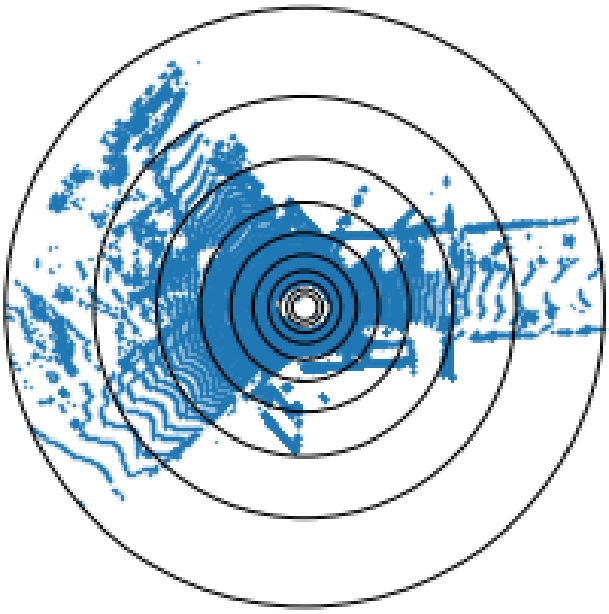
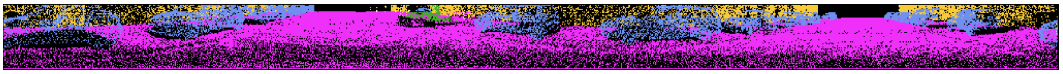


Figure 2: In black: spit of the space to compute the distribution of the points. In blue: an example of the SemanticKITTI dataset. Here the distribution is seen from top-view, the circles are actually spheres in 3D. Between two spheres, N points will be samples. The list of Ns is the following, starting from the space between the smallest and the second smallest sphere: [6159, 6883, 20252, 24437, 16630, 15982, 14253, 10516, 7144].



(a) Point cloud not aligned with the street before projection.

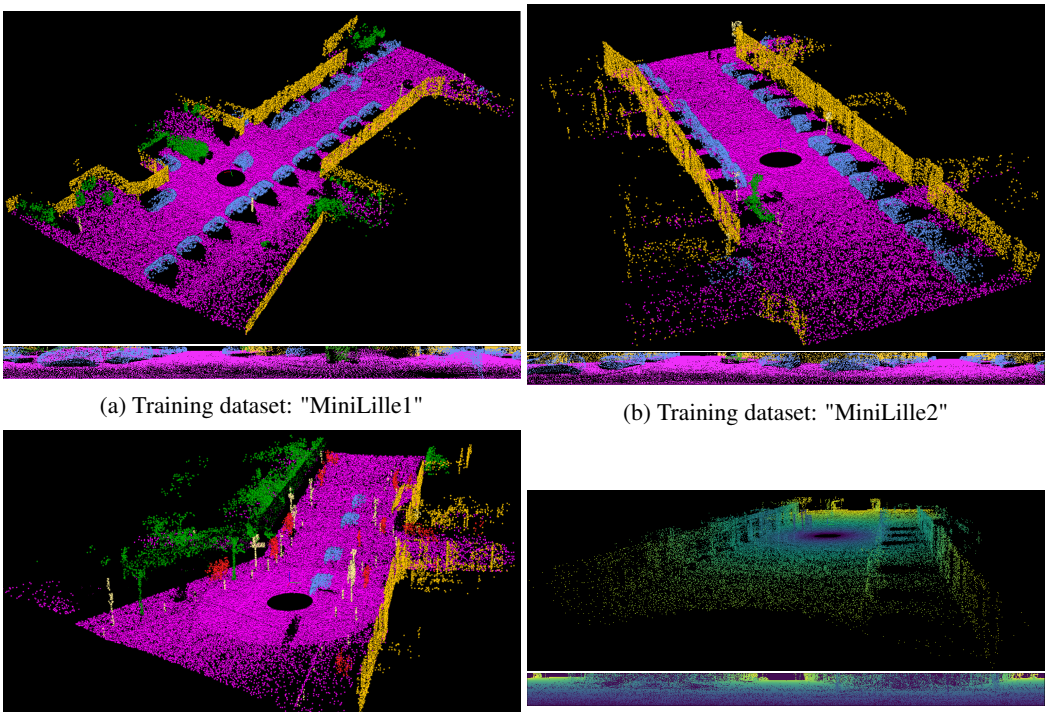
(b) Point cloud aligned with the street before projection.

(c) Example of projection from SemanticKITTI dataset.

Figure 3: Importance of rotating the point cloud.

The ground level can be computed similarly as before.

Last but not least, the outdoor point cloud has to be rotated to make sure that it is aligned with the orientation of the Velodyne of the SemanticKITTI dataset. Indeed, since the car was always moving forward, the point cloud has to be rotated in the street direction. This makes the front view of the vehicule appear in the middle of the spherical projection. This is crucial since the spherical projection changes the shape of the objects so it has to be in the same way as the one the RangeNet++ is used to. On the experiments presented in this report, the rotation was computed using the admissible centers. Figure 3 shows two examples, one where the center of the projection is in the middle (sub-figure 4b) and another one where the center of the projection is a car (sub-figure 4a). We can see how the shape of the car is different. Additionally, an example of projection of KITTI dataset is provided (sub-figure 4d).



(a) Training dataset: "MiniLille1"

(b) Training dataset: "MiniLille2"

(c) Training dataset: "MiniParis1"

(d) Testing dataset: "MiniDijon9"

Figure 4: Example of generation of virtual scans for each dataset. For each sub-figure a view of the 3D scan is represented above of the projection.

2.3 Transforming the outdoor point cloud

Once all the characteristics have been computed from the subsections 2.1 and 2.2, we choose at random an admissible center and change the mean of the dataset towards this center, fixing the ground level as the one from the SemanticKITTI dataset. Then, we apply the rotation and clip the points that are too far, too close and too high from the center. Finally, the number of points has to be reduce by subsampling the points from the point cloud such that the remaining points follow the learnt spacial distribution.

The strenght of this method is that it allows us to generate an infinite number of virtual scans. Indeed, the virtual center is chosen randomly. Furthermore, outdoor point clouds usually have way more than 120 000 points so the subsampling creates unique scans even if the center is the same.

Figure 4 shows examples of the generation of virtual scans from the outdoor point clouds of the dataset given in the benchmark of 3D Point Cloud classification for Master NPM3D Course.

3 Training

3.1 Outdoor point cloud dataset

For all the experiments in this report, the outdoor point cloud dataset that has been used is the one provided by the challenge available at `https://npm3d.fr/benchmark-for-master-course-on-3d-point-clouds`. It consists in three training point clouds and one testing point cloud. Each point is labeled according to 5 categories: unlabeled, car, person, road, building, vegetation and pole. Since there are less categories than for the SemanticKITTI dataset, I build a mapping from one category to the other.

	Average	Cars	Pedestrians	Ground	Building	Vegetation	Poles
IoU	80.6	97.8	63.3	95.6	85.9	85.1	55.7

Table 1: Train Intersection over Union (IoU) in percentage.

A major difference between the two datasets is that the SemanticKITTI dataset provides an additionnal field: "remission". In order to replace it, I computed the verticality of the normal associated with each point.

3.2 Fine-tuning

The fine-tuning was made on 5000 virtual scans over 5 epochs. Note that the number of virtual scans could be increased if the hardware devices allow it. The chosen architecture is the one described in subsection 1.2, with W = 1024 which is the highest number available for the pretrained networks on the repository (`https://github.com/PRBonn/lidar-bonnetal`) associated with the paper RangeNet++. The training was done using a NVIDIA Tesla P100 with 25 GB of RAM during thirty minutes. The greatest batch size before filling the RAM entirely was 8. Table 1 shows the results achieved after the training on the outdoor point cloud with the most diversity (all the classes were present in the point cloud).

4 Experiment

The goal is the segment the test point cloud of the challenge introduced in subsection 3.1. As this test point cloud contains points that are higher than the maximum height of the SemanticKITTI dataset, I trained a random forest classifier on the features: "x", "y", "z", "verticality", "linearity", "planarity" and "sphericity". The four last ones are obtained by computing the PCA of the neighbors of each point. The neighborhood is defined with a ball of radius 50 cm. This allowed me to have some predictions for the points that are out of the scope of the virtual scans.

For the rest of the points, I generated 1000 scans over the test point cloud. This produced more than one prediction per 3D points. To aggregate them, I used a weighted average on the predictions. It is working like a voting system except that the vote of some points is more taken into account. The weights were of the form $e^{-\frac{10 \cdot r}{max\_distance}}$ where r is the range i.e. the distance of the point to the virtual Velodyne. This choice was made because RangeNet++ is more accurate on the closest points by looking at the training set.

Another way of aggregating the prediction would be to store the probabilities given by RangeNet++ for each virtual scans. After summing them up, we would take the class with the highest probability. This methods comes from the paper of Y. Ali et al. [1]. It was not considered here since in the paper only two predictions are aggregated but in our case it is 1000 so it would require a lot of memory for a result that does not take the relative position of the virtual Velodyne into account.

Lastly, a filtering over the entire testing point cloud is performed using a KNN with a radius of 50 cm.

5 Results

The submission of the experiment ended at the third place of the ranking as table 2 shows it. The most interesting result is the IoU of the "car" category that ended 35.5 points of percentage above the second best result. This shows the power of the RangeNet++ algorithm. It is not surprising to see that the result on the "car" category is good since the algorithm has been craft for autonomous driving. A particular attention has been taken for this category. The fact that the results on the "building" and "vegetation" categories are not satisfying is due to the points out of scope of the RangeNet++ network where the random forest achieves poor results.

	Average	Cars	Pedestrians	Ground	Building	Vegetation	Poles
IoU	53.8	85.5	5.5	97.7	77.1	38.1	19.0
Rank	3	1	3	1	3	3	3

Table 2: Test Intersection over Union (IoU) in percentage.

6 Conclusion

The method presented in this report makes the transfer of the RangeNet++ from LiDAR scans to outdoor point clouds possible. It enables satisfying performances on categories like "cars" in a short fine tuning time.

Further work could lean on the points that are out of scope for RangeNet++. A first option would be to use another architecture than RangeNet++ so that the height dimension can be increased. This means that fine-tuning would not be possible anymore. Another option would be to use more features as proposed by [6] and multiple scales [3].

References

[1] Yara Ali Alnaggar, Mohamed Afifi, Karim Amer, and Mohamed ElHelw. Multi projection fusion for real-time semantic segmentation of 3d lidar point clouds. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1800–1809, 2021.

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

[3] Timo Hackel, Jan D Wegner, and Konrad Schindler. Fast semantic segmentation of 3d point clouds with strongly varying density. *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences*, 3:177–184, 2016.

[4] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[6] Martin Weinmann, Boris Jutzi, Stefan Hinz, and Clément Mallet. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105:286–304, 2015.