Computers & Graphics 93 (2020) 13–24

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on 3DOR 2020

SHREC 2020: 3D point cloud semantic segmentation for street scenes

Tao Ku[a,*], Remco C. Veltkamp[a], Bas Boom[b], David Duque-Arias[c], Santiago Velasco-Forero[c], Jean-Emmanuel Deschaud[d], Francois Goulette[d], Beatriz Marcotegui[c], Sebastián Ortega[d], Agustín Trujillo[d], José Pablo Suárez[d], José Miguel Santana[d], Cristian Ramírez[d], Kiran Akadas[e], Shankar Gangisetty[e]

[a] Department of Information and Computing Sciences, Utrecht University, Utrecht 3584CC, the Netherlands
[b] Cyclomedia Technology, the Netherlands
[c] Mines ParisTech, PSL Research University, 60 Blvd Saint Michel, Paris 75272, France
[d] Centro de Tecnologías de la Imagen (CTIM) – Universidad de Las Palmas de Gran Canaria, Las Palmas de Gran Canaria 35017, Spain
[e] KLE Technological University, Hubballi 580031, India

ARTICLE INFO

Article history:
Received 15 May 2020
Revised 31 July 2020
Accepted 27 September 2020
Available online 2 October 2020

Keywords:
SHREC 2020
3D point cloud
Semantic segmentation
Benchmark

ABSTRACT

Scene understanding of large-scale 3D point clouds of an outer space is still a challenging task. Compared with simulated 3D point clouds, the raw data from LiDAR scanners consist of tremendous points returned from all possible reflective objects and they are usually non-uniformly distributed. Therefore, its cost-effective to develop a solution for learning from raw large-scale 3D point clouds. In this track, we provide large-scale 3D point clouds of street scenes for the semantic segmentation task. The data set consists of 80 samples with 60 for training and 20 for testing. Each sample with over 2 million points represents a street scene and includes a couple of objects. There are five meaningful classes: building, car, ground, pole and vegetation. We aim at localizing and segmenting semantic objects from these large-scale 3D point clouds. Four groups contributed their results with different methods. The results show that learning-based methods are the trend and one of them achieves the best performance on both Overall Accuracy and mean Intersection over Union. Next to the learning-based methods, the combination of hand-crafted detectors are also reliable and rank second among comparison algorithms.

1. Introduction

With the application of LiDAR sensors in many areas, such as autonomous driving and augmented reality, efficient understanding of large-scale 3D point clouds is a challenging task. Many LiDAR datasets [1–4] have been released as benchmarks for various 3D applications. Thee benchmarks usually provide cleaned 3D point clouds with well-annoated labels ignoring the problems of noises, sampling pattern difference and weak reflective strength. However, it is still difficult to develop an effective and efficient solution. There are two main reasons: One is that the raw data from LiDAR scanners consist of a tremendous amount points returned from all possible reflective objects, which brings in heavy computation cost. Another reason is that the points are usually non-uniformly distributed, unstructured and unordered, which means that it is not easy to apply mature 2D deep convolutional networks.

Since the input point clouds are orderless and usually sparse, feature learning methods for point clouds can be divided into three categories in terms of the format of input point clouds: projection based, voxel based and point-set based. Projection-based methods [5,6] try to solve 3D problems via 2D approaches. Thus, 3D point clouds are projected to 2D images through different view-points. Then, common 2D CNNs are capable to deal with 3D tasks. Voxel based methods [7,8] usually map raw points to voxel representations first, then 3D CNNs are adopted to extract features. However, such methods do not naturally capture the inherent structure information of 3D point clouds. Therefore, point-set based methods are proposed to learn representative features directly from point

* Corresponding author.
E-mail addresses: t.ku@uu.nl (T. Ku), r.c.veltkamp@uu.nl (R.C. Veltkamp), bboom@cyclomedia.com (B. Boom), david.duque@mines-paristech.fr (D. Duque-Arias), santiago.velasco@mines-paristech.fr (S. Velasco-Forero), jean-emmanuel.deschaud@mines-paristech.fr (J.-E. Deschaud), francois.goulette@mines-paristech.fr (F. Goulette), beatriz.marcotegui@mines-paristech.fr (B. Marcotegui), sebastian.ortegatrujillo@gmail.com (S. Ortega), agustin.trujillo@ulpgc.es (A. Trujillo), josepablo.suarez@ulpgc.es (J.P. Suárez), josemiguelsantananunez@gmail.com (J.M. Santana), tenmacrs@gmail.com (C. Ramírez), akadask@gmail.com (K. Akadas), shankar@kletech.ac.in (S. Gangisetty).

https://doi.org/10.1016/j.cag.2020.09.006

Table 1
Summary of train and test datasets of our Street3D benchmark.

Split	Num. of Samples	Num. of Per-sample Points	Total Points	Class Ratio					
				Undefined	Building	Car	Ground	Pole	Vegetation
Train	60	2.40M–4.67M	217M	8.37%	17.05%	2.82%	54.65%	0.47%	16.64%
Test	20	2.30M–4.50M	72.8M	6.08%	22.07%	2.19%	53.96%	0.78%	14.92%

Table 2
Comparison of some outdoor 3D LiDAR datasets. For LiDAR type, TLS means Terrestrial Laser Scanning, MLS stands for Mobile Laser Scanning and ALS, Aerial Laser Scanning.

Name	LiDAR Type	Num. of Points	Num. of Scans	Num. of Classes	Annotation
Semantic3D [1]	TLS	1660M	30	8	point-wise
KITTI [2]	MLS	1799M	14,999	3	bounding box
Paris-Lille-3D [3]	MLS	143M	7	50 (training) and 10 (test)	point-wise
DublinCity [4]	ALS	260M	1	3 (coarse) and 8 (refined)	point-wise
Street3D (Ours)	MLS	290M	80	5	point-wise

coordinates. PointNet [9] is the pioneer learning point-wise features for 3D classification and segmentation tasks. Further work [10–12] have shown their power in processing and learning from 3D point clouds. However, due to the limited main memory and possible information loss from downsampling large point clouds, it's still hard for these learning-based methods to train on large point clouds without downsampling.

In this Shape Retrieval Challenge (SHREC) track on 3D Point Cloud Semantic Segmentation for Street Scenes, we compare different frameworks to segment semantic objects from large-scale 3D point clouds. Both hand-crafted and learning-based methods are included for evaluation. The contributions are summaried as:

- We provide a large-scale 3D street-scene point cloud dataset for 3D semantic segmentation.
- We evaluate different algorithms on the dataset and help finding solutions for large-scale 3D point cloud processing.
- The results show that point-set based end-to-end learning methods can outperform hand-crafted methods.

2. Street3D benchmark

2.1. Related work

In recent years, several annotated datsets for outdoor 3D point cloud semantic segmentation have been published. Semantic3D [1], recorded with a terrestrial laser scanner (TLS), is a large outdoor 3D LiDAR dataset with over a billion points and 8 class labels. However, it has only 30 samples for training and testing, and the huge amount of points per sample (because of high precision of TLS) leads to heavy memory usage when training without downsampling. KITTI [2] offers nearly 15 thousands of scans of outdoor objects, but has only three classes (car, pedestrian and cyclist) and is labelled with bounding boxes instead of being point-wise annotated. It concerns more on 3D object detection instead of 3D semantic segmentation in large-scale scenarios. Paris-Lille-3D [3] is another worth noting 3D LiDAR dataset with 50 classes from which 10 classes are used for testing. It has over 143 million points with 7 different scans and is recorded by a mobile laser scanner (MLS). DublinCity [4] provide a manual annotation of over 260 million laser scanning points with 13 hierachical classes in one scene. It has 3 coarse level classes (building, ground and vegetation) and 10 refined classes (e.g. door, tree and sidewalk). It is one of the densest urban LiDAR dataset recorded with an aerial laser scanner (ALS). These datasets have either the problem of huge size of a single scan (Semantic3D and DublinCity) or few scans for training and testing (Paris-Lille-3D and DublinCity).

In contrast to these datasets, our Street3D dataset provides less dense but more scenes for outdoor 3D semantic segmentation tasks. It consists of a large amount of well annoated points with 5 classes and 80 different street scenes. The size of each scene makes a trade-off between precision and productivity. A comparison is summaried in Table 2.

2.2. Dataset

The point cloud data is provided by Cyclomedia Technology, where we selected 80 3D point cloud for street scene and manually labelled them. The point clouds are recorded using both a panoramic image capturing device that capture every 5 meter together with a Velodyne HDL-32 Lidar sensor that performs continues capturing on the car. The recorded pointcloud is meshed, where for each pixel in the panoramic images, we compute the intersection (by ray tracing) with the dense mesh from the original pointcloud. The point of intersection provides us both color from the panoramic images capturing device together with position information.

Each point cloud represents a street scene and contains a group of objects. We used the open source software Cloud Compare [13] to manually label point clouds. Objects are labelled into 5 meaningful classes and an extra 'undefined' class. They are: 'Building', 'Car', 'Ground', 'Pole' and 'Vegetation'. Only the five meaningful classes will be evaluated. The 80 3D point clouds are randomly divided into train and test sets with 60 for training and 20 for testing. For each point cloud, there are over two million points. Summary information of the dataset is in Table 1. Examples of the dataset are visualized in Fig. 1. The dataset is made publicly available at https://kutao207.github.io.

2.3. Evaluation

We adopt the evaluation criteria that have been widely applied in 3D semantic segmentation tasks, that is the Overall Accuracy (OA) and mean Intersection over Union (mIoU).

Generally, OA reports the percent of points in the data set which are correctly classified, and mIoU is the average of per-class IoU. The IoU of class i is defined as:

$$\mathrm{IoU}_i = \frac{\mathrm{TP}_i}{\mathrm{GT}_i + \mathrm{Pred}_i - \mathrm{TP}_i} \tag{1}$$

where TP_i, GT_i, Pred_i denote the correctly classified number of points, the ground truth point number, and the predicted point number for class i, respectively.

3. Methods

In this part, we introduce four novel methods as well as a baseline method – PointNet++ for our 3D point cloud semantic segmentation task. The architectures of these four novel methods are

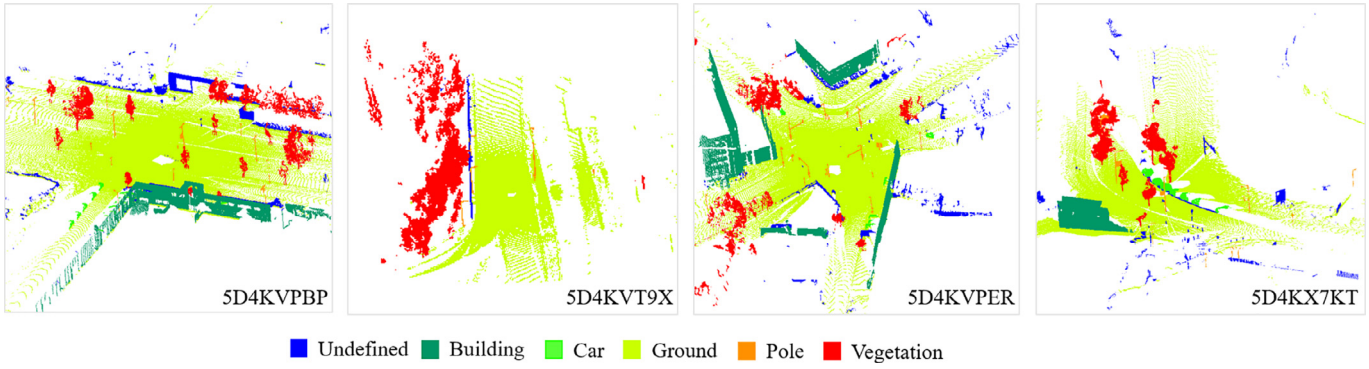

Fig. 1. Examples of the Street3D datasets. The bottom right strings are the filename of each scene.

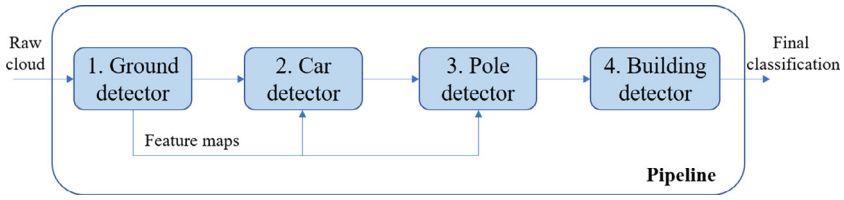

Fig. 2. Multi-stage architecture of the vehicle-borne point cloud segmentation process.

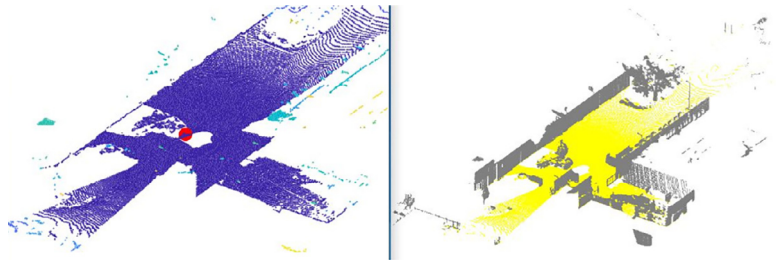

Fig. 3. Minimum height clusters and sensor position (left figure, sensor in red) vs final ground cells (right). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

varied. P4UCC is a non-learning framework of a four-staged pipline which aims to classify the 5 classes progressively. Spherical DZNet projects 3D point clouds to 2D images and train a CNN model for semantic segmentation. Then the results are back-projected to 3D world and interpolated for 3D point clouds. ResGANet adopts graphs to encode the geometric information of 3D point clouds and use a residual graph attentional network to train an end-to-end model to predict point cloud semantic labels. GRandD-Net combines multi-scale features and Unet architecture to build a powerful model for the 3D point cloud semantic segmentation task. More details are described as follows.

3.1. P4UCC: Progressive 4-staged Urban Cloud Classifier

This method is contributed by authors Sebastián Ortega, Agustín Trujillo, José Pablo Suárez, José Miguel Santana, and Cristian Ramírez. Our proposal is a four-staged pipeline that aims to progressively classify ground, car, pole, building, and vegetation points in vehicle-borne point clouds. Its general architecture can be seen in Fig. 2. Each stage aims to identify only a certain class. The first one is dedicated to ground point detection and also generates pixel-wise features which are useful for subsequent stages. Stage 2 looks for the detection of car points, Stage 3 filters pole points and Stage 4 splits the remaining points between building and vegetation classes. The code is available at https://github.com/ChanoOT/P4UCC.

3.1.1. Ground detector

Ground detection starts by dividing the cloud horizontally in a grid with squared cells of $1m^2$. Pixel features are computed using the points in each cell: minimum height, H [14]; point accumulation, A; and geometric features such as omnivariance, O; and eigensum, E [15]. Minimum height is used to select the ground candidates: all points in the cloud with height lower than $H_{ij} + 0.25$. Those points are then grouped using a hierarchical approach, euclidean distance and cutoff of 0.5 m. Cluster centroids and their inclination angles with respect to the LiDAR sensor are computed. In case the sensor position is not known, the mean of point positions of the maximum A_{ij} can be used as approximation. This can be seen in Fig. 3. The point clusters deemed ground are the ones with an inclination angle lower than the one of the vehicle in which the sensor is mounted. For the SHREC benchmark, it was approximated to 1.5°. The cluster with the largest point size is also added.

3.1.2. Car detector

Points from cells with $E_{ij} < 1.25$ are considered for car detection. These points are grouped using the same hierarchical clusters of Stage 1. A cell descriptor is calculated as $\Delta X^* \Delta Y$. A planarity descriptor [16] is also calculated per cluster. A cluster is considered to represent a car when its area is shorter than $40m^2$ and the planarity is greater than 0.2. Examples of this can be seen in Fig. 4, right.

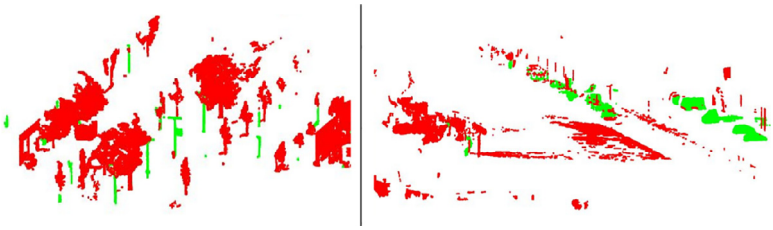

Fig. 4. Examples of pole (left, green) and car (right, green) clusters in two clouds. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 5. Final result in 5D4KVPG4 cloud. Building, pole, vegetation, car and ground are represented with red, blue, green, magenta and yellow colors, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

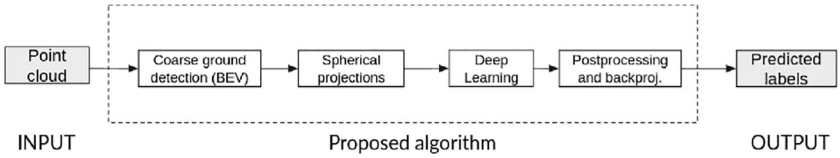

Fig. 6. Workflow of automatic point cloud segmentation.

3.1.3. Pole detector

Points from cells with $O_{ij} < 2$ are considered for pole detection and clustered. Curvature change [15], dominant component, minimum relative height and deviation of the *XY* component values are computed per cluster. Point clusters are classified as poles when the dominant component is *Z*, the curvature change is lower than 0.01, the relative minimum height is lower than 1 and the deviation of the values is lower than 1 in both *X* and *Y* components. This can also be seen in Fig. 4 (left).

3.1.4. Building detector

The remaining points are grouped as well. A recursive algorithm for extraction of vertical planes based in MLESAC [17] is applied to each cluster. Clusters with a majority of points in a vertical plane and with $\Delta Z > 2.5$m are considered to represent buildings. Otherwise, they are classified as vegetation. This leads to the final result that can be seen in Fig. 5.

3.2. Spherical DZNet

This method is contributed by authors David Duque-Arias, Santiago Velasco-Forero, Jean-Emmanuel Deschaud, Francois Goulette and Beatriz Marcotegui. We propose an automatic algorithm for point cloud segmentation using only geometrical data [*x*, *y*, *z*]. The proposed methodology is based on 2D projections. The main process relies on a spherical projection from the sensor point of view while bird eye view (BEV) projection is used for ground detection. The FAISS library [18] is used in order to reduce computational cost at several steps in the process. The proposed algorithm is presented in Fig. 6 and it is mainly divided into four steps: (1) coarse ground detection from Bird-Eye View (BEV) by means of quasi-flat zones; (2) spherical projections and feature extraction; (3) train Deep Learning model with 2D images; (4) postprocessing and backprojection 2D predictions to 3D using a kNN classfier (with $k = 3$). The code is available at https://github.com/daduquea/sphericalDZ.

3.2.1. Coarse ground detection

High class imbalance is a typical issue in point cloud segmentation. In order to reduce it, ground points are identified first. A simple quasi-flat zones algorithm from the BEV projection [19] is used for this purpose, assuming that ground elevation varies smoothly.

3.2.2. Segmentation in 2D projections

In the spherical projection, each point cloud is represented by an image of 1024×64 pixels corresponding to the azimuthal and elevation angles, respectively. A set of five features is selected in this study: *z* coordinate, depth (distance to sensor) and the normal vector in 3D coordinates denoted by (N_x, N_y, N_z). A very simple normal estimation based on depth image gradient [20] is used. Only points that have not been classified as ground are projected. If several 3D points fall in the same projected pixel, the classical approach to project the closest 3D point to the scanner at each pixel of the 2D image is used. Spherical projection has been chosen because the dataset is acquired with a terrestrial scanner. The scanner position is not given. We estimate it from the BEV as the barycentre of the empty space surrounded by the highest point density. Fig. 7 shows ground truth and computed features of '5D4KVPBP' from train set after ground extraction.

We trained a Deep Learning model using a variation of Unet architecture [21] as presented in Fig. 8. This choice was motivated by its capability to include input features at different steps of the decoding stage of the network. Specifically, the original input is directly concatenated to the features computed by the network as a shortcut connection, as is shown in Fig. 8. Including shortcuts in the model has demonstrated several advantages in image segmentation tasks [22] such as increasing convergence speed

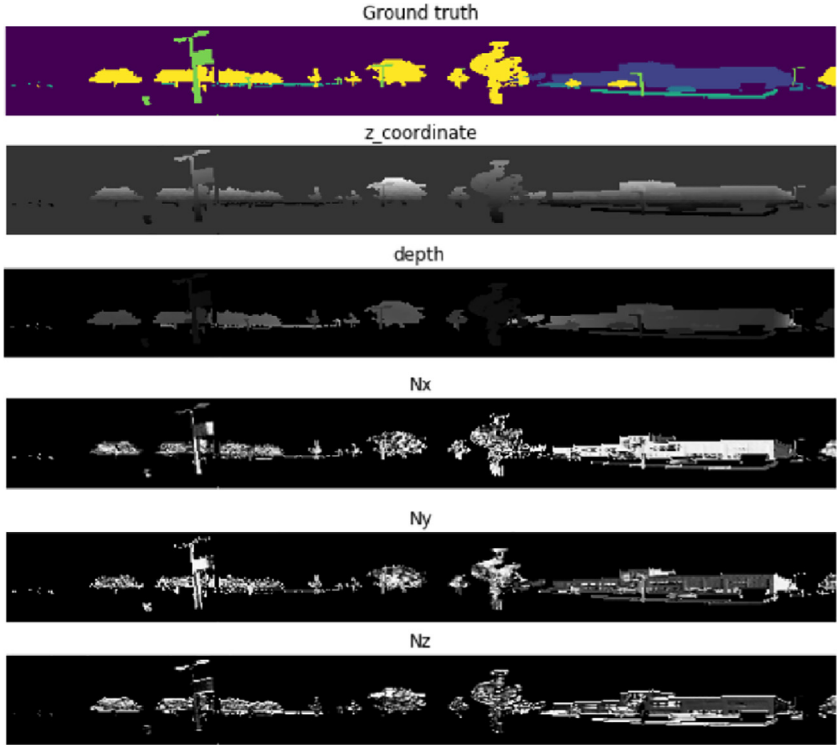

Fig. 7. Crops of spherical projections of point cloud.

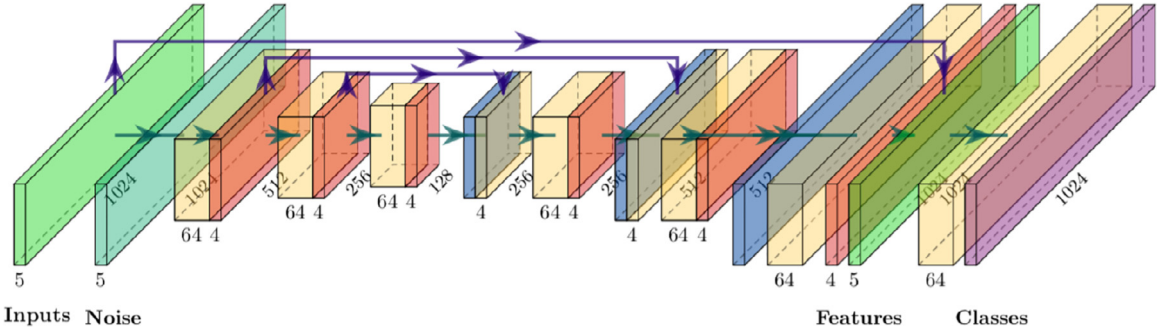

Fig. 8. Proposed Network architecture. Yellow boxes represent Conv2D layers of 5x5 with strides (1,2), red boxes Conv2D of 1x1 followed by a dropout of 0.1. Blue color means Upsampling2D layer. Violet arrows illustrate concatenation operator. Note: Original input is directly concatenated to the features computed by the network. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and stabilizing weight updates. Two strategies have been tested: (1) predict labels using trained model (The code is available at https://github.com/daduquea/sphericalDZ); (2) remove last layer of the model and predict labels using kNN classifier. Results were slightly better and faster in the second case.

As the model is trained in 2D, it is required to backproject predicted labels to point clouds. In a similar way as proposed by [23], we perform kNN to label every 3D point that is not projected in 2D image. We experimentally found that best results are obtained with k equal to three.

3.2.3. Postprocessing

In order to improve the classification of some points, we propose an automatic postprocessing step to correct some of the predicted labels using mathematical morphology. We performed the postprocessing in three steps:

Poles: Poles are thin structures in front of other larger structures. Thus, they appear as regional minima in the depth projection. Those minima are added to the detected poles. Then, only structures with a geodesic elongation [24] larger than 20 are preserved.

Building: In the BEV, structures higher than 2.5 meters and with an elongation larger than 20 are considered as buildings.

Small regions: Regions classified as vegetation or building, with less than 100 pixels in the spherical view are given the class label of the large surrounding neighbors.

3.3. ResGANet: residual graph attentional networks

This method is contributed by authors Tao Ku, Remco C. Veltkamp and Bas Boom. We introduce the Residual Graph Attentional Networks (ResGANet) for the 3D point cloud semantic segmentation task. The ResGANet takes the semantic segmentation task as a supervised learning task. We use the 60 point clouds with semantic labels in training dataset to train and validate our model. As shown in Fig. 9, the whole pipeline is composed of

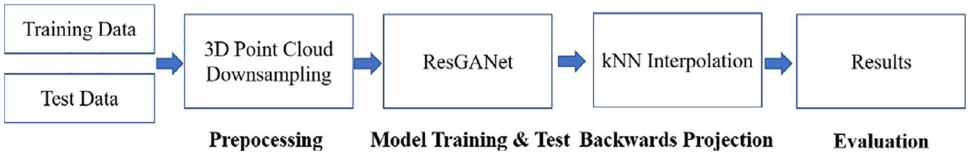

Fig. 9. Pipeline of the proposed ResGANet method.

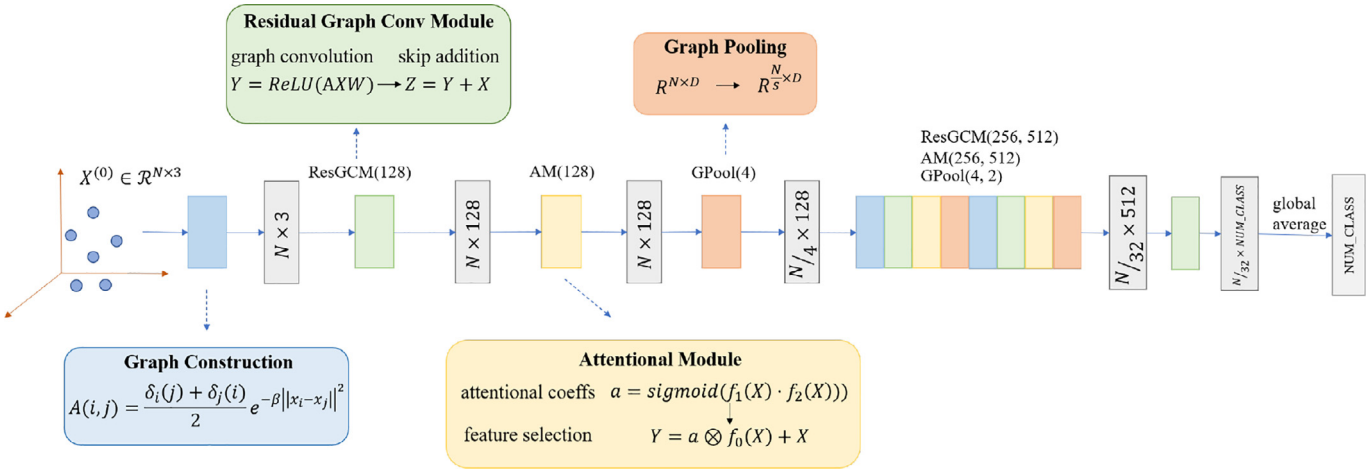

Fig. 10. Overall architecture of the proposed Residual Graph Attention Networks (ResGANet). ResGANet consists four main components: graph construction, residual graph module, attention module and graph pooling. The number after ResGCM and ATM denotes the filter number while the number after graph pooling means the pooling stride.

three parts: preprocessing, model training and evaluation, and backprojection. As our model training is based on the proposed Residual Graph Attentional Networks (ResGANet) and the input is in a fixed point number, we need to have a voxel subsampling first and then sampling the point cloud to a fixed size. After training and evaluation, we also need to back-project the valuation results on downsampled point clouds to the original ones. The code is available at https://github.com/keruast/ResGANet.

3.3.1. Preprocessing

In this part, we downsample the large 3D point clouds to smaller sizes. We apply a voxel subsampling method [12] with voxel size set to 0.1m so that dense areas can be forced to be sparse while still contain enough information. Then, we adopt a farthest point sampling [9] algorithm to sample the reduced point clouds to a fixed point number 8096.

3.3.2. ResGANet architecture

Our propose Residual Graph Attentional Networks to learn representative features directly from point coordinates. It consists of four main parts, namely, Graph Construction, Residual Graph Convolution Module, Graph Pooling and Attentional Module. The overall architecture is shown in Fig. 10.

Graph construction. The graph construction module aims at creating graph representations over point clouds. Given a 3D point cloud with N points, matrix $X \in \mathbb{R}^{N\times 3}$ denotes the input coordinates. Though there are many methods to construct a graph, in this paper, we define a graph using k-nearest neighbor (kNN) algorithms. Namely, for each point in a point cloud, we query the k nearest neighbors and connect the point with these neighbors. Considering that our graph is undirected, but kNN searching cannot ensure the graph is symmetrical, we define the adjacency as follows,

$$A(i,j) = \frac{\delta_j(i) + \delta_i(j)}{2} \exp^{-\beta||\mathbf{x}_i - \mathbf{x}_j||^2} \tag{2}$$

where $\delta_i(j)$ is an indicator function indicating whether node j is in the k-nearest neighbors of node i. If node j is in the k-nearest neighbors of node i, $\delta_i(j) = 1$, otherwise, $\delta_i(j) = 0$. $\mathbf{x}_i$ denotes the feature vector of node i. In our approach, we set the kNN query number as 16 throughout our experiments.

Residual graph convolution module. Similar to the residual block in ResNet [25], we define our residual graph convolution module by two graph convolution units and a skip connection from the input to the output addition. We adopt the graph convolution concept in [26] as

$$Y = \mathcal{A}XW \tag{3}$$

where $X \in \mathbb{R}^{N\times C}$ is the input features, $Y \in \mathbb{R}^{N\times D}$ is the filtered output, $W \in \mathbb{R}^{C\times D}$ is the weight matrix. Notice that the $\mathcal{A}$ also encodes a $(K+1)$-scalar weight variables.

Further, the residual block is defined as

$$Y_{res} = \mathcal{G}(\mathcal{G}(X, A, d), A, d) + X \tag{4}$$

where X denotes the feature input, A is the corresponding adjacency matrix and d is the degree which denotes the highest power of the dajacency matrix. $\mathcal{G}(\cdot)$ is the graph convolution function.

Graph pooling. We use kNN max pooling to define our graph pooling. Given the point cloud coordinates $X_0 \in \mathbb{R}^{N_0\times 3}$, in order to preserve the structure of point clouds, farthest point sampling [9] is used to iteratively choose the desired N_1 points. Based on that, we obtain the reduced point coordinates $X_1 \in \mathbb{R}^{N_1\times 3}$ and the corresponding index in X_0. Suppose the input feature is $F_0 \in \mathbb{R}^{N_0\times D}$. Then, for each point in the reduced points, kNN is used to find the K nearest features in F_0 such that we obtain a feature matrix of size $N_1 \times K \times D$. Finally, max pooling operation is applied to get the new feature of each selected point. An illustration of graph pooling is shown in Fig. 11.

Note that graph pooling changes the point cloud structure. Thus, the adjacency matrix in afterwards graph convolutions needs to be updated by pooled points' coordinates.

Attentional module. As attention has been successfully applied in many areas, the proposed graph attention module focuses on gathering information from important points. That is, the attention module tries to model point relations by finding out the importance of one point over all other points, and strengthening the

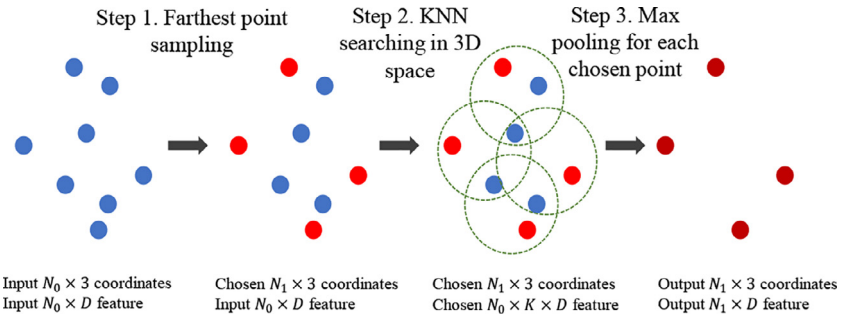

Fig. 11. The illustration of graph pooling. The goal is pooling input $N_0 \times 3$ points with $N_1 \times D$ feature to $N_1 \times 3$ points with $N_1 \times D$ feature. Each point in the figure represents a feature vector. The red ones denotes points chosen by farthest point sampling algorithm while blue ones denotes rest unchosen points. Each green circle after step 2 indicates the kNN search results. Final dark red ones are max pooling output for each chosen point feature together with their K-nearest point features. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

feature of strong points while suppressing weak point representations.

The attention module has three branches. Two branches are to calculate pairwise similarity, then the similarity output is fed into a sigmoid function to generate the attention matrix. Each element of the matrix indicates the relative influence one point has on another point compared to all other points. Instead of directly applying pairwise vector production, we first use two graph convolution layers to dynamically learning different representations of the identical input feature. Then, in case of rapid gradient descending, we use a residual-like operation by adding the attention output with the original input feature to get the final output. In this way, we re-balance the features of different points by considering each point's influence on others.

Given a feature matrix of X with N points, our attention output can be modeled as:

$$Y_{att} = \text{sigmoid}\left(\frac{f_1(X) f_2^T(X)}{\sqrt{D}}\right) f_0(X) + X \tag{5}$$

where $f_0(\cdot)$, $f_1(\cdot)$ and $f_2(\cdot)$ are feature extractors, and D is the number of feature channels.

3.3.3. Interpolation

In this part, we need to backward project the downsampled predicted labels to the raw point clouds. We use kNN to do the interpolation. Give the downsampled point coordinates with predicted labels and raw point coordinates, we determine the raw per-point label with $k = 3$ for the kNN search.

3.4. GRanD-Net: Grid subsampling, RandLA-Net and Dilated Convolutions

This method is contributed by authors Kiran Akadas and Shankar Gangisetty. We propose a deep-learning based 3D semantic segmentation of point clouds inspired from the works of RandLA-Net [12] known as GRanD-Net shown in Fig. 12. We improve the RandLA-Net [12] by considering point features in terms of (x, y, z) coordinates, subsample using Grid subsampling and adopting dilated convolutions [27] to gain better efficiency for semantic segmentation of large-scale 3D point clouds. The code is available at https://github.com/KiranAkadas/GRanDNet.

3.4.1. Data preparation

To efficiently process data of such large-scale 3D scenes, we down-sample the point clouds using grid subsampling of KPConv [11] with a fixed grid size 0.06m. In order to get back the original number of semantic labels from the predictions, we index projections for up-sampling the point clouds. The training dataset is augmented by scaling and rotation.

3.4.2. Data loading

To load the data in batches, we generate the data flow for each batch. For a given batch size n and the steps in each epoch s, $(n \times s)$ point clouds are reserved for each epoch. To avoid ordered learning by the GRanD-Net model, we feed the data randomly. The k-Nearest Neighbours (kNN) algorithm is used with a pre-defined set of k neighbours being selected of all the sub-sampled points. If the sampled points are less than the given predefined k points, we pick the points with replacement. To prepare a batch of point clouds, we generate the neighbour indices for every point in a point cloud. These are used to get the relative point features. We then randomly sample 25% of points to be reduced in the next phase while down-sampling and simultaneously track the indices for up-sampling. The pooling indices are obtained using kNN search for every sampled point.

3.4.3. Training

We train the sampled point clouds over several batches of data. The loaded point clouds of dimensions (N, d_{in}), where N is the number of points in the point cloud and d_{in} is the number of features associated with each point in the point cloud, are processed using the dilated residual blocks (DRBs) shown in Fig. 12. Each of the DRB includes multiple units of local spatial encoding and attentive pooling stacks. The DRBs are connected through skip-connections as proposed in RandLA-Net [12]. Unlike normal convolutions used in RandLA-Net, we use dilated/atrous convolutions [27] to implement the DRB in order to increase the receptivity of the filters without affecting the resolution and gain better efficiency shown in Fig. 13. The dilated convolutions incorporate multi-scale features, essential for semantic segmentation. These dilated convolutions make our proposed GRanD-Net faster and more efficient since we are increasing the area of filter coverage without increasing the parameters and affecting the feature learning, thus reducing the number of convolutions. The local spatial encoding in DRB uses the centre points and their kNN neighbours to encode the point cloud using relative positional information. At each step, we apply random sampling with DRB using the points we loaded earlier to reduce the size of input point cloud to 25%. The attentive pooling is used as a replacement to max-pooling in order to compute the attention score for every feature which is further aggregated to avoid loss of information and learn important local information. We use GeLU [28] as the activation layer in our GRanD-Net model for better learning of non-linear features. GeLU prevents strong negative activations which may affect the model. The curvature and non-monotonicity of GeLU is used to learn complex functions much better compared to ReLU and leaky ReLU. The output of the stacked DRBs is up-sampled and passed through multi-layer perceptrons (MLP) followed by fully connected layers shown in Fig. 12. The use of skip-connections and MLP while up-sampling

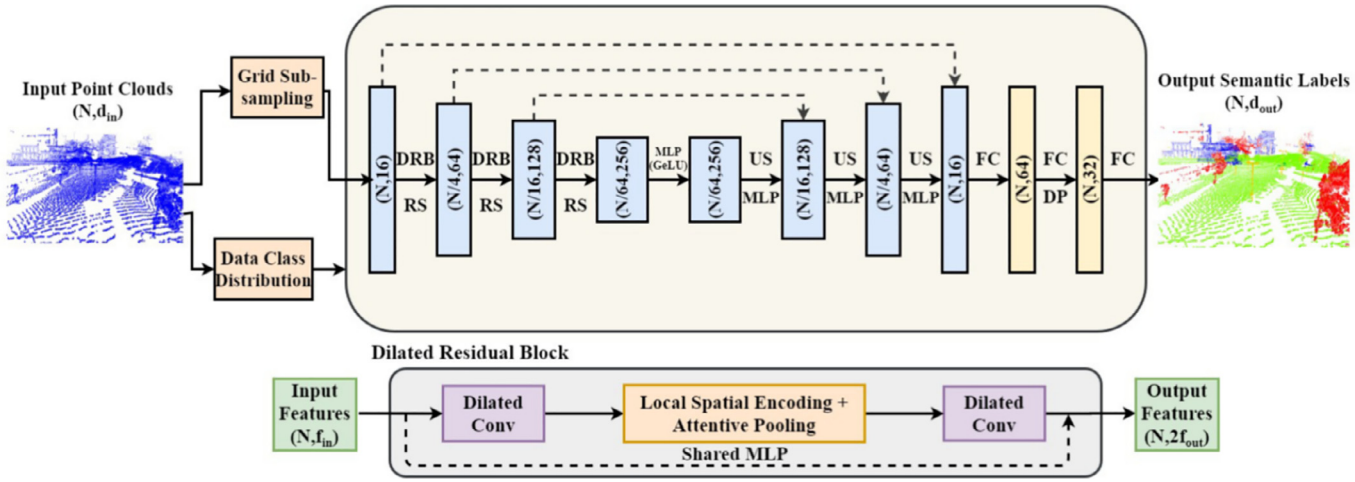

Fig. 12. The proposed architecture of GRanD-Net. The abbreviations followed in the architecture are DRB: Dilated residual block, RS: Random sampling, US: Up-sampling, FC: Fully connected layer, DP: Dropout.

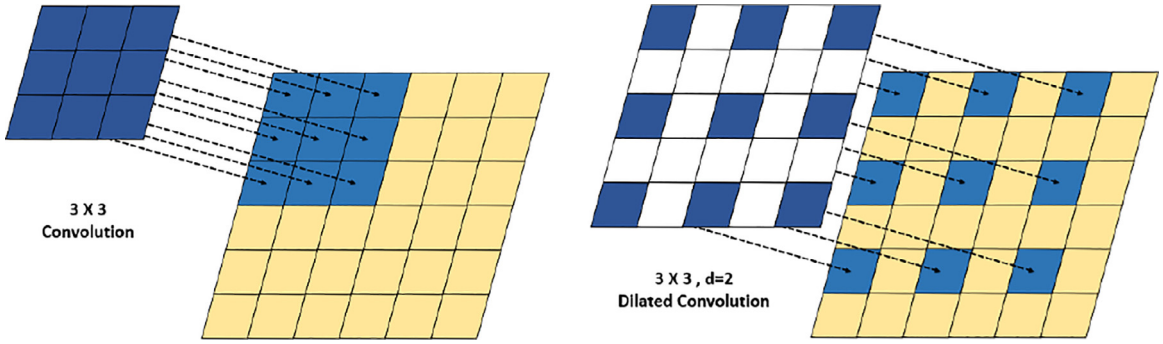

Fig. 13. Convolutions: (left) 3×3 regular convolution mostly used in CNNs. (right) 3×3 dilated convolution with d=2 covering an area of 5×5 is used in GRanD-Net resulting in lesser strides and parameters.

ensures that the labelling is accurate. Our approach follows an all-inclusive up-sampling approach that refines labels gradually unlike simple interpolation, which would result in a single label for a group of points ignoring the demarcation of classes. The use of interpolation would further require the use of a post-processing technique to refine the output. Finally, the predicted semantic labels for every point are obtained as the output of the model with dimensions (N, d_{out}), where d_{out} is the number of classes in the dataset.

3.4.4. Experimental settings

We train our proposed GRanD-Net model for 50 epochs with a train-validation split of 3:1 for the 5 classes leaving the undefined class. A four layered network is used with feature sizes of 16, 64, 128, and 256 while training. We train GRanD-Net using the Adam optimizer with a learning rate of 0.01 and a decay rate of 0.05. A grid size of 0.06 is fixed for grid-subsampling while training and we select $k = 16$ nearest neighbours to be queried. To train our model, we sample a fixed number (N) of 65,536 points from each point cloud as the input and use a batch size of 4 with 500 steps per epoch. The best performing GRanD-Net model is frozen with a mean intersection over union (mIoU) of 84.11%. The frozen model is used to predict segments for the 20 test point clouds that contain a total of 72,753,747 points. The resulting mIoU is 86.4% with an overall accuracy (OA) of 97.83% for 5 classes shown in Table 3.

3.5. PointNet++

We choose PointNet++ [29] as our baseline method for the evaluation. PointNet++ takes point coordinates as input and is able to learn deep features efficeiently and robustly. It is a very popular base algorithm for many 3D point cloud related tasks.

The implementation for semantic segmentation task is available via https://github.com/intel-isl/Open3D-PointNet2-Semantic3D.

4. Results and discussion

4.1. Experimental results

The evaluation results are shown in Table 3. Overall accuracy (OA) and mean Intersection over Union (mIoU) are evaluated for all classes, and for each class, the IoU metric is also calculated. We also plot the confusion matrix of each algorithm in Fig. 16 to help investigate the weakness of each model.

There are 5 methods under evaluation: Baseline (PointNet++), P4UCC, Spherical DZNet, ResGANet and GRanD-Net. P4UCC is based on a combination of 4 hand-crafted detectors, the other four methods are learning-based. In these four learning-based methods, Spherical DZNet is a two-stage approach with a ground detector and a projection based deep learning architecture for training and predicting, while the other three take point coordinates as input and learning representations directly from points in an end-to-end way.

Table 3
Evaluation results of our Street3D benchmark.

Method	OA (%)	mIoU (%)	Building	Car	Ground	Pole	Vegetation
Baseline	91.30	66.39	82.52	40.13	89.10	39.46	80.72
P4UCC	94.13	72.25	84.35	60.51	96.46	40.18	79.75
Spherical DZNet	93.89	67.30	83.16	49.93	96.46	27.52	79.41
ResGANet	93.55	71.39	82.96	57.66	94.40	32.94	88.98
GRanD-Net	**97.83**	**86.40**	**93.66**	**83.92**	**98.10**	**61.79**	**94.55**

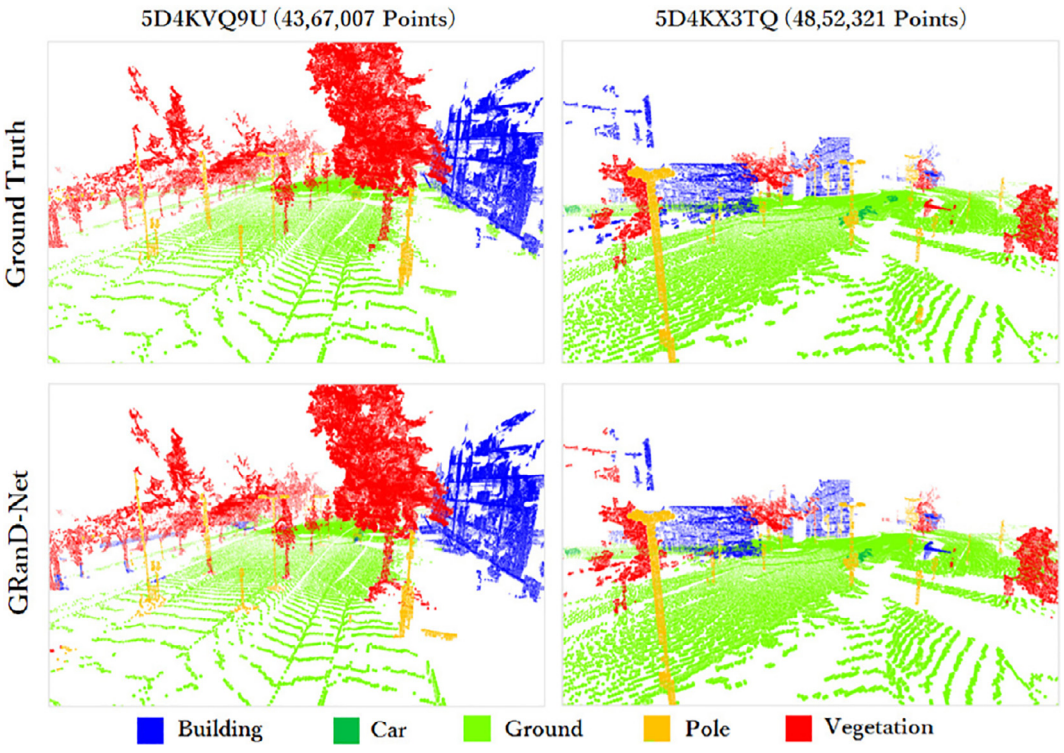

Fig. 14. Qualitative results of GRanD-Net on the test set (5D4KVQ9U and 5D4KX3TQ point clouds).

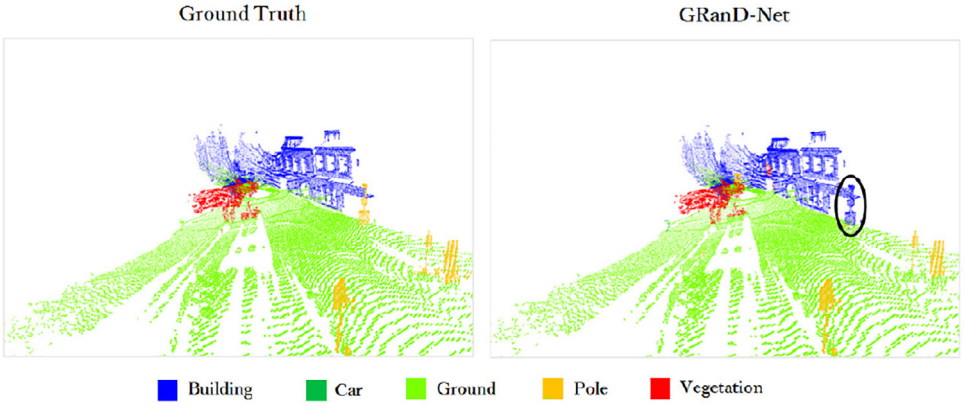

Fig. 15. Qualitative results of GRanD-Net on the test dataset. The black circle shows the pole mislabelled as building.

4.2. Computation complexity and timing

In Table 4, we show the number of trainable parameters and inference time as criterias for computation complexity and timing, since these 5 methods are run on different platforms and each has very different architecture and input format. We list the time as a reference.

4.3. Discussion

As shown in Table 3 and Fig. 16, we have the following observations:

- From obtained results with Spherical DZNet, we have identified some drawbacks related to spherical projections: (1) objects size strongly depends on distance to sensor; (2) high variation of objects shape according to sensor perspective. Both of

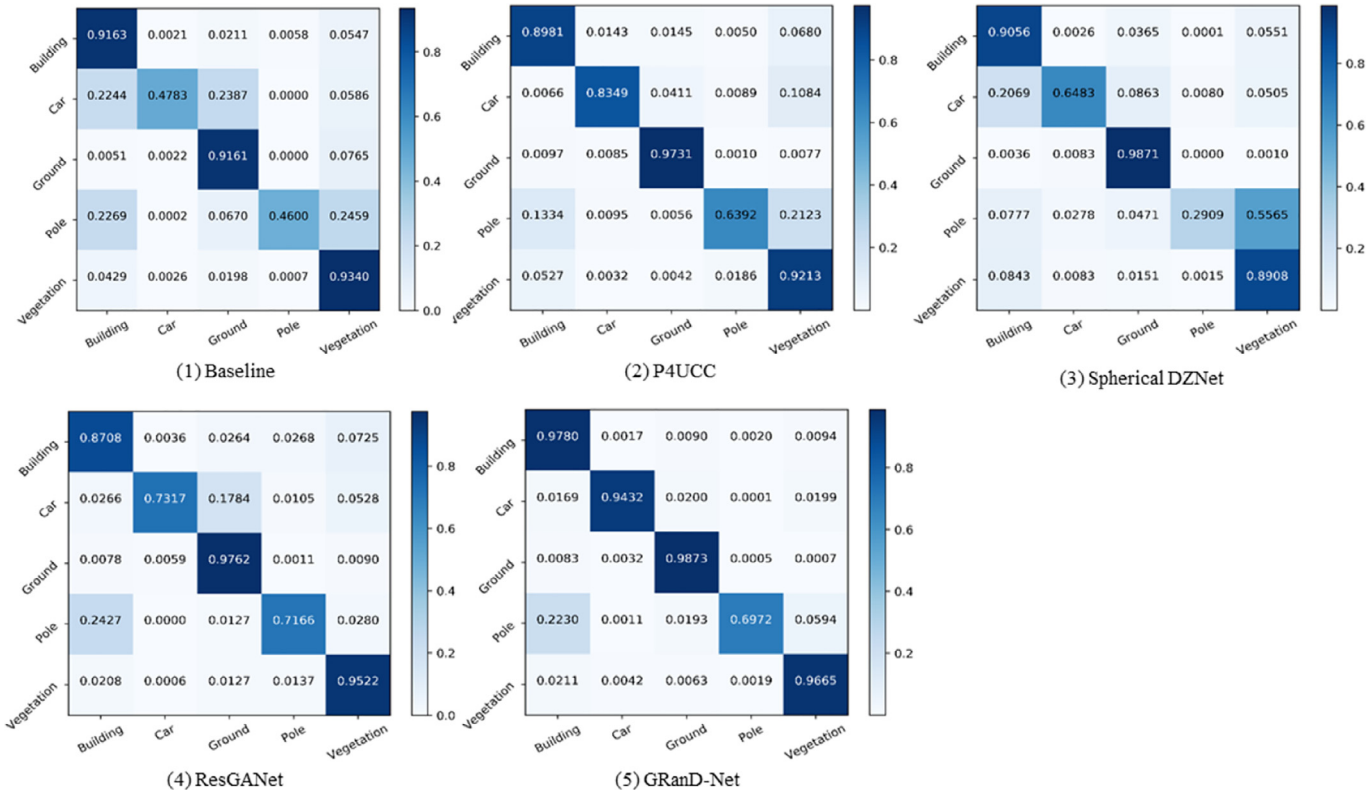

Fig. 16. Confusion matrix for all the five comparison algorithms. The *x*-axis represents the predicted labels while the y-axis denotes the groundtruth labels. In this figure, each subfigure shows clearly the classification ability of each algorithm. We normalized each row of the confusion matrix so that it is intuitive to show the probability of correct predictions and incorrect predictions.

Table 4
Computation cost for participated methods.

Method	Platform	Number of trainable parameters	Seconds per million points for inference	Other time
P4UCC	CPU Intel i5 dual-core, 8GB RAM, Matlab2018b	-	-	17.91 seconds per million points
Spherical DZNet	CPU Intel Xeon E-2186G, 32GB RAM, GPU Nvidia Titan X 12 GB	88.8K	-	Spherical projection: 0.15s, 2D training one epoch: 3.8s, 2D prediction: 1.15s, 2D to 3D: 0.81s, kNN search: 50s
Baseline	CPU Intel i7-8700, 16GB RAM, GPU Nvidia GTX-1080ti 11 GB	0.97M	768.13s	-
ResGANet	CPU Intel i7-8700, 16GB RAM, GPU Nvidia GTX-1080ti 11 GB	1.76M	850.31s	-
GRanD-Net	Trained on Google Cloud Platform, CPU - 4 vCPU, 15 GB Memory (Powered by Intel Skylake), GPU-NVIDIA Tesla T4 16GB	0.99M	219.37s	-

them may affect the capability of the model to learn. In addition, we identified a particularly low score in pole class. We think it is mainly due to the presence of traffic sign of point-cloud 5D4L1RW5. Few traffic signs are present in the dataset. Moreover, this one is close to the scanner and has over 150 thousand 3D points.

- It is worth noting that P4UCC is the only non-learning-based approach but still outperforms 3 learning-based methods (Baseline, Spherical DZNet and ResGANet) by using four detectors to identify point categories progressively. It shows that hand-crafted method is still reliable and may outperform learning-based methods.
- GRanD-Net achieves superior performance on four of the classes, except pole. We observe that the resulting IoU of ground, vegetation, building, and car classes are segmented accurately as the dataset distribution in these classes is high and learnt better. We also performed a qualitative analysis of GRanD-Net model on the test set shown in Fig. 14. Visual inspection shows that our model performance is good and close to ground truth. As the pole class IoU is low to figure out the misclassification we plotted the confusion matrix for the GRanD-Net shown in Fig. 16 (5). Based on the confusion matrix and visual inspection, we observe that few instances of the pole class are mislabelled as building shown in Fig. 15, due to their proximity to the building points and also the insufficiency of the pole training points.
- It is obvious that 'Building' and 'Ground' are easy to be correctly classified while 'Car' and 'Pole' are hard to be segmented. Considering the proportion of 'Car' and 'Pole' in the train/test set is 2.82%/2.19% and 0.47%/0.78%, all four learning-based meth-

ods cannot segment minority classes well. Non-learning-based P4UCC performs better on minority classes.
- We also observe that all three point-set learning-based methods (Baseline, ResGANet and GRanD-Net) have to downsample the large point clouds before model training due to the limit of hardware computation power. It may more or less result in information loss and influence the segmentation performance.

It is obvious that the end-to-end learning approaches are the trend to tackle with the 3D point cloud semantic segmentation task. They usually have powerful ability to segment certain classes, however they may also suffer from imbalanced data. The traditional hand-crafted method may not outperforms learning-based methods on certain classes but when the data is imbalanced, they can show their robustness and reliability.

Another issue worth noting is the limited classes of our Street3D dataset. For the 3D semantic segmentation task, it is sufficient to have 5 classes. However, it may not be enough for specific tasks. For example, 'road' is an important category for autonomous driving application. However, Street3D has 'ground' instead of 'ground'. It is not easy to recover road from only 3D 'ground' points. But our Street3D can still contribute to training and validating deep learning networks for autonomous driving application. The 4 learning-based methods in this paper can be easily transferred to or finetuned on a dataset with a 'road' label as what have usually been done in 2D CNN applications.

5. Conclusions

In conclusion, this track has drawn attention on 3D point cloud semantic segmentation for street scenes using multiple approaches. We provide a street scene dataset composed of 80 scans which are well annotated with five class labels. We introduce four novel and different methodologies that outperform a state-of-the-art deep learning based method (PointNet++) for the new Street3D benchmark. As there is a non-learning method (P4UCC) which also outperforms learning-based approaches, it shows that well-designed feature descriptors for the classification could have more importance in the segmentation than the learned features, specifically with unbalanced data. The four methods have focused not only on different architectures but also on different features (covariance-based features in P4UCC, multi-scale features in GRanD-Net, graph representations in ResGANet, spherical descriptors in Spherical DZNet), which opens the possibility of combining some of the descriptors from each discussed proposal in a new architecture. The task is still challenging as small data size classes (Car and Pole) are still hard to be classified. More work need to be done for solving the performance imbalance over classes.

Declaration of Competing Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

CRediT authorship contribution statement

Tao Ku: Methodology, Software, Writing - original draft, Data curation. **Remco C. Veltkamp:** Supervision. **Bas Boom:** Data curation. **David Duque-Arias:** Methodology, Software, Writing - original draft. **Santiago Velasco-Forero:** Methodology, Software, Writing - original draft. **Jean-Emmanuel Deschaud:** Methodology, Software, Writing - original draft. **Francois Goulette:** Methodology, Software, Writing - original draft. **Beatriz Marcotegui:** Methodology, Software, Writing - original draft. **Sebastián Ortega:** Methodology, Software, Writing - original draft. **Agustín Trujillo:** Methodology, Software, Writing - original draft. **José Pablo Suárez:** Methodology, Software, Writing - original draft. **José Miguel Santana:** Methodology, Software, Writing - original draft. **Cristian Ramírez:** Methodology, Software, Writing - original draft. **Kiran Akadas:** Methodology, Software, Writing - original draft. **Shankar Gangisetty:** Methodology, Software, Writing - original draft.

Acknowledgement

The work "Spherical DZNet" was partially funded by REPLICA FUI 24 project and ARMINES.

References

[1] Hackel T, Savinov N, Ladicky L, Wegner JD, Schindler K, Pollefeys M. Semantic3d.net: a new large-scale point cloud classification benchmark. CoRR 2017. http://arxiv.org/abs/1704.03847.
[2] Geiger A, Lenz P, Urtasun R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In: Proceedings of the 2012 IEEE conference on computer vision and pattern recognition, Providence, RI, USA, June 16-21, 2012. IEEE Computer Society; 2012. p. 3354–61. doi:10.1109/CVPR.2012.6248074.
[3] Roynard X, Deschaud J, Goulette F. Paris-lille-3d: a point cloud dataset for urban scene segmentation and classification. In: Proceedings of the 2018 IEEE conference on computer vision and pattern recognition workshops, CVPR workshops 2018, Salt Lake City, UT, USA, June 18-22, 2018. IEEE Computer Society; 2018. p. 2027–30. doi:10.1109/CVPRW.2018.00272.
[4] Zolanvari SMI, Ruano S, Rana A, Cummins A, da Silva RE, Rahbar M, et al. Dublincity: annotated lidar point cloud and its applications. In: Proceedings of the thirtieth British machine vision conference 2019, BMVC 2019, Cardiff, UK, September 9-12, 2019. BMVA Press; 2019. p. 44. https://bmvc2019.org/wp-content/uploads/papers/0644-paper.pdf.
[5] Boulch A, Saux BL, Audebert N. Unstructured point cloud semantic labeling using deep segmentation networks. In: Pratikakis I, Dupont F, Ovsjanikov M, editors. Eurographics workshop on 3D object retrieval, 3DOR 2017, Lyon, France, April 23-24, 2017. Eurographics Association; 2017.
[6] Chen X, Ma H, Wan J, Li B, Xia T. Multi-view 3D object detection network for autonomous driving. In: Proceedings of the 2017 ieee conference on computer vision and pattern recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017; 2017. p. 6526–34. doi:10.1109/CVPR.2017.691.
[7] Maturana D, Scherer S. Voxnet: a 3D convolutional neural network for real-time object recognition. In: Proceedings of the 2015 ieee/rsj international conference on intelligent robots and systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015; 2015. p. 922–8. doi:10.1109/IROS.2015.7353481.
[8] Tchapmi LP, Choy CB, Armeni I, Gwak J, Savarese S. Segcloud: Semantic segmentation of 3D point clouds. In: Proceedings of the 2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017. IEEE Computer Society; 2017. p. 537–47. doi:10.1109/3DV.2017.00067.
[9] Qi CR, Su H, Mo K, Guibas LJ. Pointnet: Deep learning on point sets for 3D classification and segmentation. In: Proceedings of the 2017 IEEE conference on computer vision and pattern recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017; 2017a. p. 77–85. doi:10.1109/CVPR.2017.16.
[10] Landrieu L, Simonovsky M. Large-scale point cloud semantic segmentation with superpoint graphs. In: Proceedings of the 2018 ieee conference on computer vision and pattern recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. IEEE Computer Society; 2018. p. 4558–67. doi:10.1109/CVPR.2018.00479.
[11] Thomas H, Qi CR, Deschaud J, Marcotegui B, Goulette F, Guibas LJ. Kpconv: flexible and deformable convolution for point clouds. In: Proceedings of the 2019 IEEE/CVF international conference on computer vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019. IEEE; 2019. p. 6410–19. doi:10.1109/ICCV.2019.00651.
[12] Hu Q, Yang B, Xie L, Rosa S, Guo Y, Wang Z, et al. Randla-net: efficient semantic segmentation of large-scale point clouds Proceedings of the IEEE conference on computer vision and pattern recognition; 2020.
[13] CloudCompare. 3D point cloud and mesh processing software open source project. http://www.cloudcompare.org. 2020.
[14] Ortega S, Trujillo A, Santana J, Suárez J. An image-based method to classify power line scenes in lidar point clouds. In: Proceedings of the twelfth international symposium on tools and methods of competitive engineering, Las Palmas de Gran Canarias, Spain; 2018. p. 7–11.
[15] Blomley R, Weinmann M, Leitloff J, Jutzi B. Shape distribution features for point cloud analysis-a geometric histogram approach on multiple scales. ISPRS Ann Photogramm Remote Sens Spatial Inf Sci 2014;2(3):9.
[16] Zhao R, Pang M, Wang J. Classifying airborne lidar point clouds via deep features learned by a multi-scale convolutional neural network. Int J Geograph Inf Sci 2018;32(5):960–79. doi:10.1080/13658816.2018.1431840.
[17] Torr PHS, Zisserman A. MLESAC: a new robust estimator with application to estimating image geometry. Comput Vis Image Underst 2000;78(1):138–56. doi:10.1006/cviu.1999.0832.

[18] Johnson J, Douze M, Jégou H. Billion-scale similarity search with gpus. CoRR 2017. http://arxiv.org/abs/1702.08734.
[19] Hernández J, Marcotegui B. Point cloud segmentation towards urban ground modeling. In: 2009 Joint Urban Remote Sensing Event. IEEE; 2009. p. 1–5.
[20] Nakagawa Y, Uchiyama H, Nagahara H, Taniguchi R. Estimating surface normals with depth image gradients for fast and accurate registration. In: Brown MS, Kosecká J, Theobalt C, editors. Proceedings of the 2015 international conference on 3D vision, 3DV 2015, Lyon, France, October 19-22, 2015. IEEE Computer Society; 2015. p. 640–7. doi:10.1109/3DV.2015.80.
[21] Ronneberger O, Fischer P, Brox T. U-net: convolutional networks for biomedical image segmentation. In: Navab N, Hornegger J, III WMW, Frangi AF, editors. Proceedings of the medical image computing and computer-assisted intervention – MICCAI 2015 – eighteenth international conference Munich, Germany, October 5–9, 2015, Proceedings, Part III. Lecture Notes in Computer Science, 9351. Springer; 2015. p. 234–41. doi:10.1007/978-3-319-24574-4_28.
[22] Drozdzal M, Vorontsov E, Chartrand G, Kadoury S, Pal C. The importance of skip connections in biomedical image segmentation. In: Carneiro G, Mateus D, Peter L, Bradley AP, Tavares JMRS, Belagiannis V, et al., editors. Deep learning and data labeling for medical applications – first international workshop, LABELS 2016, and second international workshop, DLMIA 2016, held in conjunction with MICCAI 2016, Athens, Greece, October 21, 2016, Proceedings. Lecture Notes in Computer Science, 10008; 2016. p. 179–87. doi:10.1007/978-3-319-46976-8_19.
[23] Milioto A, Vizzo I, Behley J, Stachniss C. Rangenet ++: Fast and accurate lidar semantic segmentation. In: Proceedings of the 2019 IEEE/RSJ international conference on intelligent robots and systems, IROS 2019, Macau, SAR, China, November 3-8, 2019. IEEE; 2019. p. 4213–20. doi:10.1109/IROS40897.2019.8967762.
[24] Lantuéjoul C, Maisonneuve F. Geodesic methods in quantitative image analysis. Pattern Recogn 1984;17(2):177–87. doi:10.1016/0031-3203(84)90057-8.
[25] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the 2016 IEEE conference on computer vision and pattern recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016; 2016. p. 770–8. doi:10.1109/CVPR.2016.90.
[26] Sandryhaila A, Moura JM. Discrete signal processing on graphs. IEEE transactions on signal processing 2013;61(7):1644–56.
[27] Yu F, Koltun V. Multi-scale context aggregation by dilated convolutions. In: Bengio Y, LeCun Y, editors. Proceedings of the forth international conference on learning representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings; 2016. http://arxiv.org/abs/1511.07122.
[28] Hendrycks D., Gimpel K.. Gaussian error linear units (gelus). arXiv preprint arXiv: 1606084152016.
[29] Qi CR, Yi L, Su H, Guibas LJ. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems 30: annual conference on neural information processing systems 2017, 4–9 December 2017, Long Beach, CA, USA; 2017b. p. 5099–108.