International Journal of Applied Earth Observation and Geoinformation 118 (2023) 103208

Contents lists available at ScienceDirect

International Journal of Applied Earth Observations and Geoinformation

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

Point cloud voxel classification of aerial urban LiDAR using voxel attributes and random forest approach

Harith Aljumaily [a], Debra F. Laefer [b,*], Dolores Cuadra [c], Manuel Velasco [a]

[a] Department Computer Science and Engineering, Carlos III University of Madrid, Av. Universidad 30, 28911 Madrid, Spain
[b] Center for Urban Science and Progress and Department of Civil and Urban Engineering, Tandon School of Engineering, New York University, 370 Jay Street, Brooklyn, NY 11201, USA
[c] Department Computer Science and Statistics, King Juan Carlos University, Campus de Móstoles - Madrid, Spain

ARTICLE INFO

Keywords:
Object Classification
MapReduce
Big Data
LiDAR
DBSCAN Algorithm
RANSAC Algorithm

ABSTRACT

The opportunities now afforded by increasingly available, dense, aerial urban LiDAR point clouds (greater than100 pts/m2) are arguably stymied by their sheer size, which precludes the effective use of many tools designed for point cloud data mining and classification. This paper introduces the point cloud voxel classification (PCVC) method, an automated, two-step solution for classifying terabytes of data without overwhelming the computational infrastructure. First, the point cloud is voxelized to reduce the number of points needed to be processed sequentially. Next, descriptive voxel attributes are assigned to aid in further classification. These attributes describe the point distribution within each voxel and the voxel's geo-location. These include 5 point-descriptors (density, standard deviation, clustered points, fitted plane, and plane's angle) and 2 voxel position attributes (elevation and neighbors). A random forest algorithm is then used for final classification of the object within each voxel using four categories: ground, roof, wall, and vegetation. The proposed approach was evaluated using a 297,126,417 point dataset from a 1 km^2 area in Dublin, Ireland and 50% denser dataset of New York City of 13,912,692 points (150 m^2). PCVC's main advantage is scalability achieved through a 99 % reduction in the number of points that needed to be sequentially categorized. Additionally, PCVC demonstrated strong classification results (precision of 0.92, recall of 0.91, and F1-score of 0.92) compared to previous work on the same data set (precision of 0.82-0.91, recall 0.86-0.89, and F1-score of 0.85-0.90).

1. Introduction

Urban objects such as roadways, trees, and buildings are presently identified using a multiplicity of data sources including satellite imagery, Light Detection and Ranging (LiDAR), environmental sensors, and even crowd sourced information (Aljumaily et al. 2019). Spatial databases or some form of a Geographic Information System (GIS) has traditionally been used for their storage, manipulation, and viewing. Given the rapidly increasing size of such data sets standard, stand-alone computer resources are no longer becoming viable means for achieving the most basic functionality and usefulness of such data without turning to the complexity of distributed computing.

Additionally, collaborative work products such as Digital Surface Models (DSMs) of limited areas produced from aerial LiDAR point cloud data no longer meet community needs, in part because such workflows remove the original data points and, thus, preclude further exploitation the data's richness. Initially, DSMs were adopted to overcome data sparsity. Today a different problem exists – namely, the exceedingly large size of the data sets. Typical aerial LiDAR missions consist of billions of tuples (3D) points requiring terabytes, if not petabytes, of storage. For example, the United States' first national aerial LiDAR survey has already released more than a trillion points, with more to still be acquired (USGS 2020). Processing data sets of this size is further challenged by the data's dimensionality. Each tuple had 3 coordinates (x, y, and z), a timestamp, and an intensity measurement. When imagery is collected at the same time or subsequently registered, each tuple has color indicators (typically Red-Green-Blue). There may also be affiliated flight path and sensor data. In 2015, Vu et al. (2015) charted a projection of aerial LiDAR density increase of almost an order of magnitude per decade. Notably, a single autonomous vehicle already collects terabytes

* Corresponding author.
E-mail addresses: haljumai@inf.uc3m.es (H. Aljumaily), debra.laefer@nyu.edu (D.F. Laefer), dolores.cuadra@urjc.es (D. Cuadra), velasco@ia.uc3m.es (M. Velasco).

https://doi.org/10.1016/j.jag.2023.103208
Received 21 February 2022; Received in revised form 18 January 2023; Accepted 22 January 2023
Available online 11 February 2023

of LiDAR per hour (Ramesh et al. 2019). Thus, traditional LiDAR storage and processing solutions will only increasingly struggle to support the rapidly escalating number of LiDAR users and the ever-expanding types of queries for these rapidly growing data sets. As an example, in the debate in the US congress for funding the national LiDAR survey, more than 600 use cases were identified (USGS 2020).

The sheer size of the anticipated data densities and spatial extents provide further impetus to using strategies commonly referred to as Big Data, as noted nearly a decade ago by Xue et al. (2014) for applications such disaster mitigation, high-level modeling, and map updating. Big Data platforms can offer a logical and useful choice for storage and analysis of huge volumes of data through parallel and distributed computing. To demonstrate the viability of this, a fully automatic, two-step approach is proposed herein to classify high-density, aerial LiDAR point clouds into four classes: ground, wall, roof, and vegetation. The first step partitions the urban area into equisized voxels to reduce the number of points being processed. In the second step, these voxels are assigned seven descriptors related to the point distribution and the voxel s position. A random forest algorithm is then used for classification. The significant contributions over previous work by the authors (Aljumaily et al. 2017; Aljumaily et al. 2019) are the scalability and the introduction of a means to overcome manual parameter selection with respect to selecting the maximum number of outlier points when using both the DBSCAN and RANSAC based algorithms.

The main contribution of the proposed approach is that it demonstrates the implementation of a fully automatic and significantly more scalable approach than otherwise available to support mapping the LiDAR data in a Big Data context by reducing the number of datapoints during the classification task by approximately 99%. Thus, this paper offers a two-step solution for classifying terabytes of data without overwhelming the computational infrastructure.

The remainder of this work is organized as follows: Section 2 reviews the background literature; Section 3 describes the scope and methodology; Section 4 presents validation experiments; Section 5 discusses the obtained results, and Section 6 formulates general conclusions, emphasizes the main contribution of the paper, and provides thoughts on future work.

2. Related work

Data classification for the purpose of object segmentation, extraction, and reconstruction is a well-established topic within the geomatics community (Yang et al., 2015). Approaches often rely upon point grouping of similar features or processing point-to-point data features. Ni et al. (2017) described three main strategies for LiDAR data classification with respect to granularity of the processing unit (irrespective of whether these are done with or without machine learning or deep learning techniques): (1) point-based classification (e.g. Yastikli and Cetin 2016; Guo et al., 2015), (2) segment-based classification (e.g. Yang et al., 2015; Xiang et al. 2016), and ((3) multiple-entity-based classification (Vosselman et al., 2017).

Point-based methods extract individual features for each point, apply a classifier to train with various samples and then classify the point cloud with the obtained classified model. For obtaining features for every single point, a neighborhood concept is applied that describes the 3D structure around the point. Such techniques need a parameter (usually the radius) to describe the proximity of the points related to density and structure in the dataset.

Segmentation techniques divide the point cloud into a set of geometric structures (planes, voxels) including the ground plane. Applied functions to join points in a geometric structure can be by region (e.g. Truong-Hong et al. 2013), aggregation functions (e.g. Vu et al. 2015, Aljumaily et al. 2015); or combining both approaches (e.g. Li and Sun 2018). A disadvantage to such approaches relates to reliance on only geometric structures. This can limit the effectiveness, but they are useful to calculate geometric features that mitigate dependency on neighborhood optimization methods and provide several new attributes helpful to use semantic rules (e.g. if planes then walls; if rough then trees). Furthermore, the approaches typical require calculation of eigenvectors and eigenvalues for each point in a repetitive manner and are, therefore, time-consuming.

Multiple-entity-based classification uses points and segments with other data to improve classification accuracy (Xu et al. 2014) such as joining aerial LiDAR data with terrestrial and mobile LiDAR, as well as imagery (Awrangjeb et al. 2013)] or GIS-based data [e.g. OpenStreetMap data (Aljumaily et al. 2019)]. Such approaches are not uncommon but may require overcoming data format or device incompatibilities (Previtali et al. 2014; Nikoohemat et al. 2017). As an example, multiple-entity-based classification by Vosselman et al. (2017) employed information directly obtained from the point cloud (e.g. derived orientation and neighborhood context) combined with independent domain classifiers as part of a hybrid segmentation to obtain planar segments and segments of arbitrary shapes and sizes using Conditional Random Field. The outcomes of those efforts demonstrated that segment-based classification was more effective in utilizing context information than a point-based approach (91% vs 82.8%) when using a 30 pts/m^2 dataset from Rotterdam.

Beyond those are machine learning (ML) approaches that learn about an object in context (e.g. area, surface dimensions, orientation, similarity, proximity, coplanarity, and orthogonality). These have also been used to extract regions, patterns, points, and objects (Chakrawarty et al. 2014), as well as direct classification, as recently summarized by Liu et al. (2019). Notable examples include the work by Weinmann et al. (2015) in which four components (neighborhood, feature selection, feature extraction, and classification) were used to support individual point classification. That work concluded that classification algorithms are more efficient when information associated with a point s neighbors is included. As an example of this, Qi et al. (2016) applied a neural network called PointNet that directly consumes point clouds. Once the features were selected, the data were grouped into feature vectors to train the model to predict membership class (point or segment).

Common ML approaches used with LiDAR data include K-Nearest Neighbors (KNN), SVM, Convolutional Neural Networks (CNNs) and Random Forest (RF). Notably, most CNNs provide point cloud classifications without any pre-processing as described by Liu et al. (2019). Recent work in this area includes that by Zhang et al. (2020) who employed a CNN in the extraction of a digital elevation model by including specific features of the LiDAR data. Their approach organized the point cloud data in a patch-based structure that entailed a pre-clustering of points with similar characteristics. Their proposed CNN improved models such as PointNet , although there were still inefficiencies in the application due to the complexity of the deep neural network and the high computational cost in data preprocessing and network inference. According to Lin et al. (2020) deep neural networks have made significant breakthroughs in point cloud classification and segmentation tasks, such as PointCNN. While successful for complicated feature representation, deep learning approaches require thousands of pieces of ground truth data for training, as demonstrated by Zolanvari et al. (2019) where over 2,500 h were expended to complete an 8-category classification of 260 million points (representing less than 20% of a high-density aerial data set from 2015).

Among the wide range of ML and DL approaches, Random Forest (RF) [Breiman, 2001] has been particularly widely adopted. According to Belgiu and Dragut (2016), RF is less sensitive than CNNs for LiDAR urban objects classification and does not tend to overtrain. RF was first applied for LiDAR classification in 2009 (Chehata et al., 2009). This supervised algorithm consisted of a build phase and an operational phase (Wang et al., 2019). The build phase trained the algorithm and generated the classification model, while the operational phase performed the classifications. Therefore, the training set quality is very important and usually something that is prepared manually, which is very time consuming and subject to the introduction of human error. In

the application of RF by Park and Guldmann (2019), 11 attributes were used to classify data as point-specific, footprint-relative, and point-neighborhood with the goals of building and roof classification, as well as height estimation. The overall accuracy was around 97%, which decreased to 93% in residential areas, because of the presence of shrubs, fences, and other items in the environment that added noise. Bassier et al. (2019) recently concluded that while ML methods often generalize better than heuristic ones (e.g. Soilan et al., 2020), they require extensive training data, as previously noted by Zolanvari et al. (2019). Despite this constraint, RF has shown itself to be a computationally efficient technique and able to produce higher accuracies than other approaches (e.g. Farnaaz and Jabbar, 2016) without requiring extensive training times (Park and Guldmann; 2019; Belgiu and Draguţ, 2016; Khan et al. 2010; Oshiro et al. 2012). RF has the additional benefit of being applicable to multi-class and multi-attribute classification problems (Wang et al., 2019).

The RF prediction model has been used as a ML technique in many LiDAR projects. For instance, Naidoo et. al. (2012) applied RF to classify 8 common savanna tree species using a combination of hyperspectral and LiDAR-derived structural parameters, in the form of a 7 predictor dataset resulting in an accuracy of approximately 87%. Slightly lower success rates with RF (mean precision and recall measures of 78% 85%) were obtained by Azadbakht et al. (2016) using a fivefold, cross-validation (four training splits and one validation split) when applied to urban objects of trees, vegetation (shrubs and grass), multiple roof classes, asphalt, concrete, vehicles, power lines, walls (fences) and bodies of water. In Ahmed et al. (2015), RF and multiple regression were applied separately to model canopy cover and height for three forest classes (mature, young and mature, and young). The RF models provided improved estimates relative to the multiple regression models across all classes. Bassier et al. (2019) demonstrated better performance with an RF classification when compared to other approaches including KNN, a multiceptron Neural Network (NN), and Support Vector Machine (SVM). Consequently, RF was selected herein to improve classification performance within a Big Data environment.

Specifically, this paper proposes a classification algorithm based on the creation of device-independent and format-independent, grid-based subspaces with the intent of providing more scalability than other RF implementations (e.g. Bassier et al., 2019). Unlike many state-of-the-art approaches (e.g. Park and Guldmann, 2019; Azadbakht et al. 2016), the proposed Point Cloud Voxel Classification (PCVC) approach uses attributes derived from point geometry, context geometry (for each subspace), and global attributes to achieve a model sufficiently robust to classify difficult classes such as low vegetation or trees near building facades. To overcome the limitations of complex implementation and high computational costs [as noted by Xu and Stilla (2021)], the proposed PCVC approach is introduced within a scalable, Big Data platform.

3. The proposed approach

The proposed PCVC approach employs RF to classify objects within urban, aerial point cloud data. Although point cloud data attributes typically include x-, y-, and z-coordinates, intensity, return number, number of returns, scan direction flag, edge of flight line, and classification, in the proposed approach only the x-, y-, and z-coordinates for each point were used, which vastly simplifies its implementation. This approach involves two steps (Fig. 1). Step 1 is grid-based subspace partitioning, where points are grouped in a set of 3D voxels. This strategy facilitates easier and more efficient data handling than working directly with individual points and makes the procedure less sensitive to noise, without reliance on a supervoxel or recursive voxel exploration. In contrast, in Step 2 descriptive voxel attributes are assigned for further classification. These attributes mainly describe the point distribution within each voxel along with the geolocation of the voxel. After attribute calculation, RF is applied to classify these voxels (see Section 4). These steps are further described below.

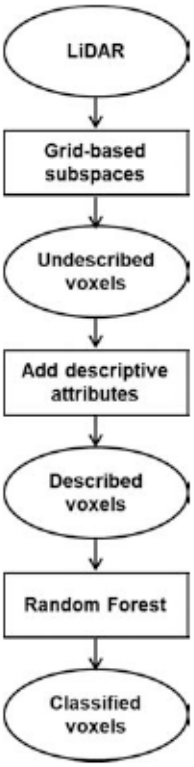
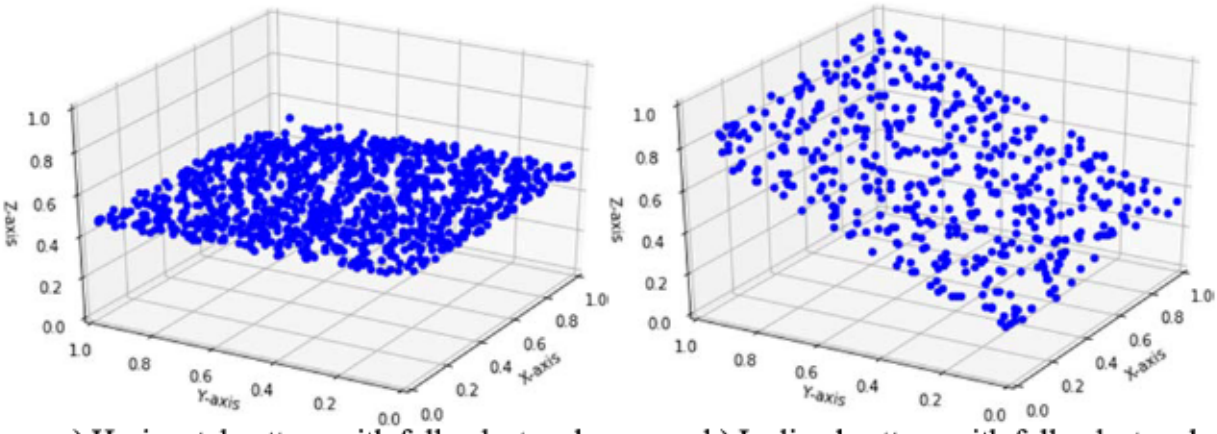


Fig. 1. Proposed approach with 3 stages of outputs: (1) undescribed voxels; (2) described voxels; and (3) classified voxels.

3.1. Step 1: Grid-Based subspace partitioning

Three-dimensional, grid-based methods for subspace partitioning significantly reduce time complexity, especially for high-dimensional datasets (Aljumaily, et al. 2019). Thus, the basic idea presented herein is to exploit the coordinates (x, y, z) of each point in the point cloud by mapping these into smaller, equisized, 3D subspaces (i.e. voxels). To achieve this, a value of D 1 m is used as the grid resolution in each direction The proposed volume is only a small fraction of the entire point cloud and smaller than most objects found in cities. In Aljumaily, et al. 2017, an experiment was done by selecting three different voxel dimensions (0.5 m, 1.0 m, and 2.0 m) of the same point cloud for object detection. The execution time using voxels with dimensions of 0.5 m were very similar to those using dimensions of 1.0 m, but a value of 1.0 m for D was recommended, because when the voxel dimension was only 0.5 m there were many voxels of low density. This is particularly problematic for distinguishing vertical surfaces from noise in aerial LiDAR data sets, as the yield of vertical data is only about one-tenth of that of the horizontal data yield as was established by both Hinks et al. (2009) and Stanley and Laefer (2021) in multiple, urban, aerial LiDAR missions. If a larger grid (e.g. 2 m) is selected, other problems arise including the accidental inclusion of multiple objects within a single voxels. This is particularly problematic in the presence of vegetation. Additionally, larger voxels result in a high number of points needing to being processed within each voxel. Consequently, more execution time is then needed.

As such, in this Step 1, consider a point cloud consisting of a set of 3D points; p_1 (x_1, y_1, z_1), p_2 (x_2, y_2, z_2), , p_n (x_n, y_n, z_n), where n is the number of points, and m is a set of voxels identical in size to each other [C_1 (X_1, Y_1, Z_1), C_2 (X_2, Y_2, Z_2), , C_m (X_m, Y_m, Z_m)]. If D 1, then p (x, y, z) C (X, Y, Z), if (X x X 1) and (Y y Y 1) and (Z z Z 1). For example, for point coordinates 315603.38, y 233870.5, z 15.38 and D 1, the point is mapped to the voxel coordinates of X 315603, Y 233870, Z 15.



a) Horizontal pattern with fully clustered points

b) Inclined pattern with fully clustered points

Fig. 2. Dense voxels (typically part of building objects or the ground).

The grid-based subspaces are achieved by using the MapReduce framework, a programming model used to support parallel computing. The approach consists of two functions: Map and Reduce, which operate using Key-Value data types. The Map function receives an input data and issues a list in the form of (K_1, V_1), where K_1 represents the voxel coordinates, which is calculated according to the previous rule, while V_1 represents the current point being processed. Next, the Reducer receives the list of the pairs (K_1, V_1) and issues a list of the pairs (K_2, V_2), where K_2 is the voxel coordinate, as a unique key in the list, while V_2 is a list of all points that belong to K_2. For more information about applying MapReduce for voxel division see Aljumaily et al. (2015).

3.2. Step 2: Assigning descriptive voxel attributes

In this step, attributes based on point distribution inside each voxel and the geographical positions of the voxel within the larger data set are determined. To begin this process, a point cloud PC is mapped to a set of voxels PC $\in \{C_1, C_2, C_3 \ldots C_M\}$, where M is the number of voxels in the PC. Each voxel C_i (1 m × 1 m × 1 m) consists of a set of points, $C_i \in \{p_1, p_2, p_3 \cdots p_N\}$, where N is the number of points in C_i. In this case, five distinctive point descriptors (density, standard deviation, clustered points, fitted plane, and plane's angle) and two voxel position attributes (voxel elevation and voxel neighbors) are determined These area described below.

a) Density (DENS).

This attribute represents the overall point density within each voxel, $DENS_i = N_i / 1^3$ where N_i is the number of points within C_i. This attribute provides an indicator of importance. For example, a voxel with only one point would likely represent noise, while a voxel containing more than 10 points would likely represent an object or part of an object.

b) Standard Deviation (STDV).

To measure the amount of point variation or dispersion, the standard deviation $STDV_i$ for each C_i is calculated. For this, let $\{(x_i, y_i, z_i)$, where $i = 1,\ldots n\}$ be a set of points of C_i. Then $STDV_i$ is calculated as follows:

$$(x_-, y_-, z_-) = \frac{1}{n}\sum_{i=1}^{n}(x_i, y_i, z_i) \quad (1)$$

From that, the standard deviation of C_i can be calculated by:

$$STDV_i = \sqrt{\frac{(x_- - x_i)^2 + \left(y_- - y_i\right)^2 + (z_- - z_i)^2}{n}} \quad (2)$$

c) Clustered points (CLUS).

There are two primary point distribution patterns: dense and sparse (Aljumaily et al. 2015). A dense voxel is defined as containing a set of successfully clustered points either horizontally or diagonally (Fig. 2) that form a highly populated sector within the voxel.

Consider C_i as a given voxel, $CLUS_i = n_i / N_i$, where n_i is the number of clustered points in C_i. If $CLUS_i = 1$, then C_i is a fully clustered or dense voxel. Normally, a dense voxel forms part of a building object (mostly roofs) or part of the ground including roads and sidewalks.

A sparse voxel contains a high percentage of dispersed points (i.e., $CLUS_i \approx 0$). These voxels tend to represent vegetation (Fig. 3a) [due to the ability of laser scanner to penetrate gaps among the foliage] or a vertical wall (Fig. 3b); low vertical point densities are an artefact of the aircraft's relative position to the building wall causing large angles of incidence, thereby limiting point acquisition on these vertical surfaces, as demonstrated by Hinks et al. (2009). Aboveground utilities, humans, automobiles in transit, and general noise are classified as sparse voxels, due to their partial representation in typical aerial point clouds. For $CLUS_i = (n_i / N_i)$, the clustered points, n_i, for each voxel need to be calculated. This is done with a DBSCAN clustering algorithm (Ester et al., 1996), which efficiently partitions clustered and dispersed spaces in a k-dimensional space by calculating the distance between points. For that, consider p_j and p_k as two points belonging to a voxel C_i, if Distance $(p_j, p_k) <$ Eps (the maximum radius threshold to delimit the neighborhood of a point p_j), where p_j is considered as a core point, and p_k is its neighbor. Otherwise, point p_k is considered as noise. Additionally, a point p_j is the core of a cluster, if at least MinPts points exist in its neighborhood region and the distance that point and the rest of the points of the cluster is less than EPS. Thus, defining the two parameters of Eps and MinPts is crucial to the proposed clustering process, because an overly large Eps will unintentionally include points that do not belong to that cluster, while an overly small Eps will exclude points that actually belong. The same problem arises with respect to MinPts. If an excessively low MinPts is selected, then low-density clusters may be included, while an overly high MinPts would unintentionally exclude some high-density clusters.

Herein, n_i = DBSCAN (C_i, Eps, MinPts) is used mainly to calculate the number of clustered and dispersed points in C_i. For example, Fig. 4 shows a dense voxel with 227 points. Of these, 184 are clustered points (blue points) and 45 points are dispersed points (red points) [i.e., $CLUS_i = 0.80$]. Importantly, the approach detects arbitrarily shaped clusters, which means that the cluster formation is independent of both the voxel's orientation and the specific details of the actual object, as well as largely independent of specific point density.

d) Fitted plane (FITT).

As previously noted, the attribute $CLUS_i$ can easily distinguish between fully clustered and fully dispersed voxels. However, there are voxels where classification is not straightforward, such as that shown in

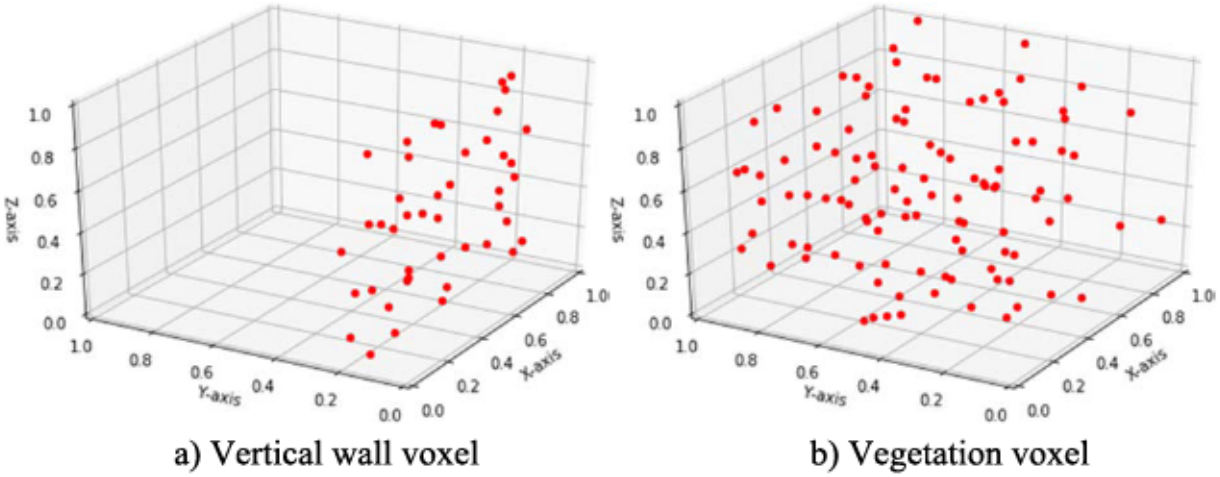


Fig. 3. Sparse voxels (typically vegetation or vertical wall).

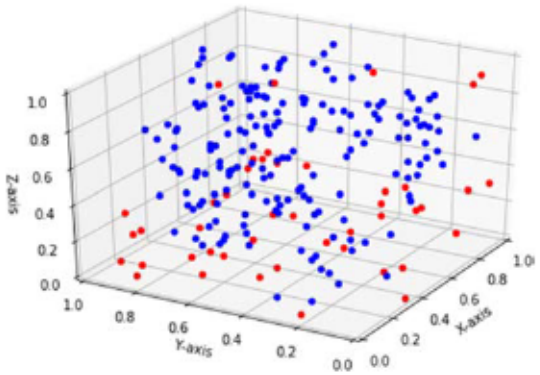


Fig. 4. Clustered points shown as blue points and disperse points as red points. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 4s, which ha a high $CLUS_i$ of 0.80 but no visible pattern. To aid in the classification of this type of voxel, a fitted plane is introduced, in the form of the RANdom SAmple Consensus (RANSAC) algorithm as proposed by Fischler and Bolles (1981). RANSAC is used to find the best plane that can be fitted through a set of 3D points within a voxel.

For that, let C_i be a given voxel, $FITT_i = n_i / N_i$, where n_i is the number of fitted points onto a plane C_i. If $FITT_i = 1$, then C_i has fully fitted points. Normally, a fully fitted voxel forms part of a building object (mostly roofs) or part of the ground. An unfitted voxel contains a high percentage of dispersed points (i.e., $FITT_i \approx 0$).

According to the RANSAC procedure, a plane is calculated using three points randomly selected from the set. Inlier and outlier points are designated with respect to a threshold distance. In this work, a threshold distance value of 0.1 m is selected. The number of iterations, k, of the algorithm is calculated according to the following formula: $k = \log(1\text{-}p)/\log(1\text{-}(1\text{-}e)^s)$, where p is the probability to find at least one good set of inlier points in k iterations; s is the minimum number of points to fit into a model; and e is the percentage of outliers.

Fig. 5 shows an example of the application of the RANSAC algorithm in this incarnation. This fitting facilitates clear distinctions between voxels. The voxel shown in Fig. 5a is the same voxel as Fig. 3a but rotated 180° to be visually clearer. Although $CLUS_i = 0$ due to its fully dispersed points, its $FITT_i = 1$, because all of its points are fitted onto a single plane. The voxel in Fig. 5a belongs to a vertical wall. The voxel shown in Fig. 5b has $CLUS_i = 0$ due to its fully dispersed points, and $FITT_i = 83/227 = 0.36$, where 83 is the number of fitted points onto a plane, and 227 is the total number of points in the voxel. The voxel in Fig. 5b belongs to the class vegetation.

e) Plane's angle (ANGL).

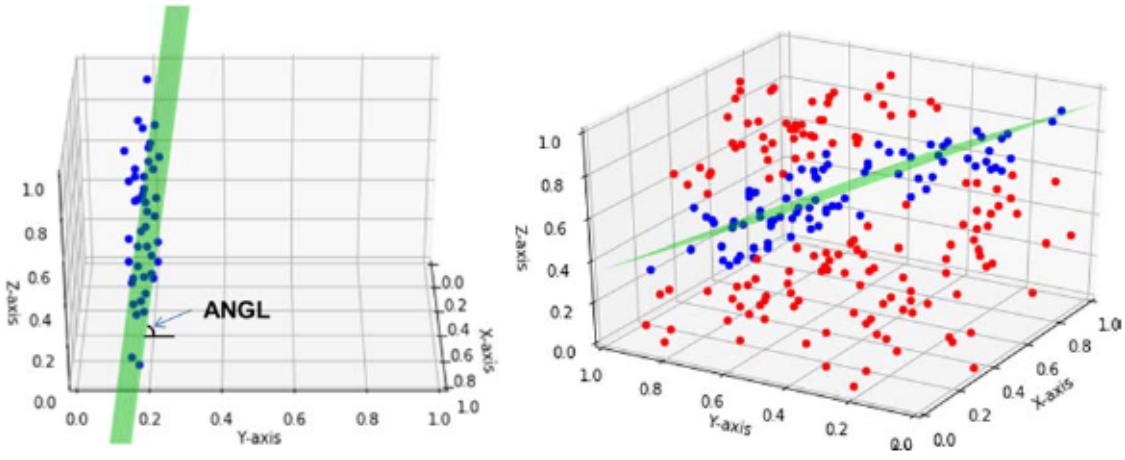


Fig. 5. Fitted plane vs unfitted plane in previously uncategorized voxels.

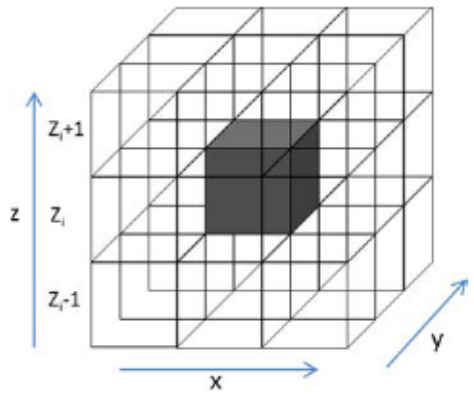


Fig. 6. Adjacent dense voxels.

This substep calculates the slope of the previously found fitted plane (Fig. 5a), which consists of a normal vector $n_1 = (a_1, b_1, c_1)$. The plane's slope is the angle between the obtained plane and a horizontal plane with a normal vector $n_2 = (0, 0, 1)$, which is calculated as per eqn (3).

$$cos(a) = \left(\frac{|c_1|}{\sqrt{a_1^2 + b_1^2 + c_1^2}} \right) \quad 3)$$

Consider voxel C_i. If cos $(a) \approx 1$, then $ANGL_i \approx 0°$. If cos $(a) \approx 0$, then $ANGL_i \approx 90°$. Normally, a perpendicular plane with $ANGL_i \approx 90°$ and $FITT_i \approx 1$ would form part of a vertical wall, while a voxel with $ANGL_i \approx 0°$ and $FITT_i \approx 1$ would form part of building object (mostly roofs) or part of the ground.

Distinguishing point distribution patterns to achieve classification is not a simple task, because multiple patterns can appear within a specific voxel type. For example, not all the bare ground voxels have horizontal point distributions, and not all the buildings roofs voxels have the same slope distribution. Thus, the voxel's position within the larger data set is considered.

f) Voxel's elevation (ELEV).

For determination of a voxel's elevation (ELEV), the originally assigned coordinates are not considered to achieve independence of any geo-referencing. Instead, the real height of each voxel is measured from the ground. The voxel's elevation can help distinguish between ground and non-ground voxels and is determined based on the following algorithm:

Divide the point cloud PC into a set of small equal point clouds $\{pc_1, pc_2, \ldots, pc_n\}$.

For each $pc_i \in PC$:

Select 3 points $\{p_1(x_1,y_1,z_1), p_2(x_2,y_2,z_2), p_3(x_3,y_3,z_3)\}$ with the smallest values of z_i in pc_i.

Find the equation of a plane $(ax + by + cz + d = 0)$ passing through $\{p_1, p_2, p_3\}$.

For each voxel $v_j \in pc_i$:

Find the elevation ELEV of v_j with the coordinates (x_j, y_j, z_j) and the plane $ax + by + cz + d = 0$, where.

$$ELEV(v_j) = \frac{|ax_j + by_j + cz_j + d|}{\sqrt{a^2 + b^2 + c^2}}$$

g) Voxel's Neighbor (NEIB).

This attribute represents the number of neighboring voxels adjacent to the target voxel (0 to 26). For a given starting voxel C_i (X_i, Y_i, Z_i) [e.g. grey voxel in Fig. 6], there exists an adjacent voxel C_j (X_j, Y_j, Z_j), where the value of X_j ranges from X_i-1 to $X_i + 1$, the value of Y_j ranges from Y_i-1 to $Y_i + 1$, and the value of Z_j ranges from Z_i-1 to $Z_i + 1$.

This attribute helps to identify voxels belonging to a vegetation object. A voxel surrounded on all sides (i.e. NEIB $\approx$ 26) can be easily understood as vegetation, while a smaller value of NEIB (i.e. $\leq$ 8) is likely indicative of a voxel on a surface (e.g. ground, roof, or wall).

4. Experiments and evaluation

The approach presented in the previous section was evaluated using a 1.0 km^2 study area in the center of Dublin, Ireland (Fig. 7). The original data of 297,126,417 points were mapped to 2,560,759 voxels, which resulted in an average of 116 points per voxel. After partitioning the point cloud into voxels of 1 m^3, the descriptive attributes of each of these voxels were calculated. Table 1 shows an example with the values of these attributes for four voxels (one per each class of ground, roof, wall, and vegetation). For example, the first voxel belongs to the class ground, consists of DENS = 324 points with standard deviation STDV = 0.26, and has all of its points clustered inside the voxel, CLUS = 1 (100%). To find the CLUS value of each voxel, values of Eps = 10 cm and MinPts = 10 were selected based on reliable usage in a previously published study (Aljumaily et al., 2017). To calculate the FITT attribute, the probability to find at least one good set of inlier points, p, was set to 0.99, the minimum number of points to fit into a model was set as s = 3 points, and the percentage of outliers, e, was assumed to be 30%, also based on

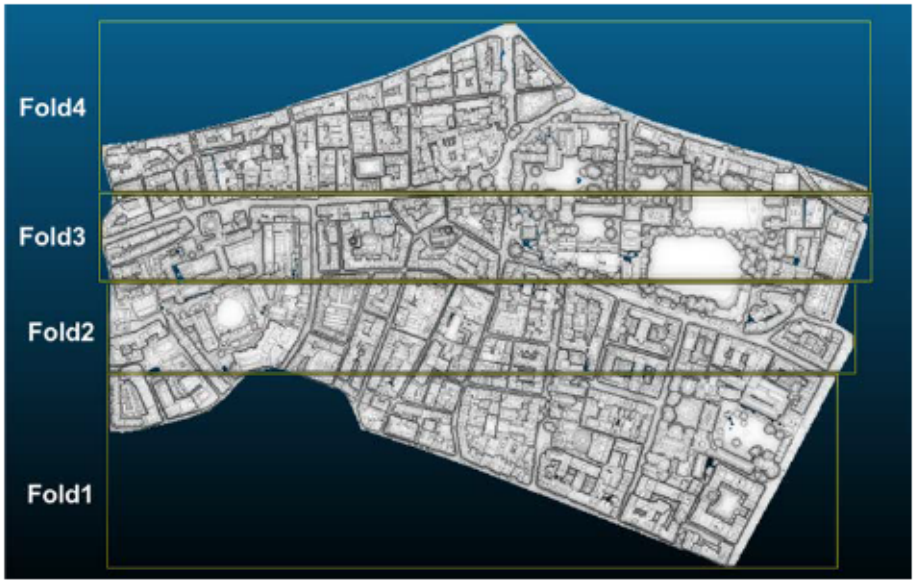


Fig. 7. Study area data.

Table 1
Example of descriptive attributes that comprise the vector (as described in the text).

x	y	Z	DENS	STDV	CLUS	FITT	ANGL*	ELEV**	NEIB***	Class	
315,603	233,870	15	324	0.26	1	1	3.85	10	8	1	Ground
315,557	233,826	14	308	0.23	0.99	0.98	4.83	7	8	2	Roof
315,425	233,845	9	40	0.313	0.22	0.8	90	3	5	3	Wall
315,557	233,825	12	103	0.275	0.77	0.83	68.92	5	25	4	Vegetation

* unit of measurement is degrees.
** unit of measurement is meters.
*** unit of measurement is the number of adjacent voxels.

previously published work (Aljumaily et al. 2019). The rest of the attributes corresponded to the angle of the plane generated by RANSAC (ANGL 3.85), the voxel s elevation (ELEV 8), and the number of the voxels neighbour (NEIB 8).

For evaluation purposes, a K-Fold Cross Validation technique was used. In that approach, the point cloud was divided into 4 folds of equal size (Fig. 7). The number and sectioning were selected to ensure that identifiable vegetation was present within each fold (Fig. 7). Of the originally mapped 2,560,759 voxels, each fold contained one-quarter (640,190) of all voxels. The distribution between categories among these folds is shown in Table 2. Of all of the voxels contained in the total study area, 14.25% were ground, 31.41% were roof, 44.67% were walls, and 9.67% were vegetation. These percentages were obtained by manual extraction for each of the categories from the original point cloud after visualization in open-source software (CloudCompare, 2015). Segmentation involved identification of each object s contour.

To quantitatively evaluate the classification results, precision, recall, and quality were calculated. Precision, which evaluates the exactness of an approach, is the ratio of the correctly categorized voxels of a specific class to the total number of voxels categorized to that class (eqn (4)). Recall, which measures the ability to extract the entire set of voxels relevant for a category (i.e. completeness), is the ratio of the extracted correctly categorized voxels to the total number of voxels in the study area (eqn (5)). The f1score evaluates the overall quality of the classification method and is calculated based on a combination of the precision and recall metrics (eqn (6)).

(4)

(5)

(6)

where True Positives (TP) are the voxels correctly included, False Positives (FP) the voxels incorrectly included, and False Negatives (FN) the voxels mistakenly excluded.

After selecting the folds, the RF algorithm was applied. For this sklearn.ensemble.RandomForestClassifier of scikit-learn.org (Python Core Team, 2019) was applied to train the algorithm. All parameters were set to the software s default settings, except that the maximum depth of trees in the forest was limited to 10 to avoid over-fitting and to help ensure computational performance. This decision was based on initial experimentation that showed increasing the depth of trees from 10 to 20 increased the execution time from 45 s to 95 s but with only a 0.1% improvement in classification using a windows PC with an Intel i7-8550U-1.80 GHz processor with 16 GB of memory.

Table 3 and Fig. 8 show the results by fold with the lowest average result being the precision (0.89) in Test 2. Looking by category (Fig. 8), the roof and wall classifications were high and consistent, with all metrics (precision, recall, f1-score) exceeding 94%. For vegetation, precision and recall were worse, although still relatively high: ranging from 0.77 to 0.87 for precision and from 0.79 to 0.84 in recall (Fig. 8). The missing vegetation voxels are likely to have been lost, in part, because of the complexity of the manual extraction, and because vegetation voxels do not have clear distribution patterns. While not perfect, the results are commendable given the extremely high variability of the building types, street types, and vegetation for a 1 km^2 urban center that incorporates more than 300 years of construction styles and includes cobbled roads, as well as paved ones.

Fig. 9 shows the confusion matrices of the four tests. While the approach was successful overall, confusion occurred principally in two classes. The first was between vegetation and walls where 2%-3 % of the wall voxels were misclassified as vegetation and 16%-21% of the vegetation voxels were misclassified as walls. The other confusion was between the ground and roof classes, where 7%-18% of the ground voxels were misclassified as roof and 3%-5 % of the roof voxels were misclassified as ground. These errors were higher than expected given the incorporation of voxel position in the training and may be ultimately overcome with more training data.

To test the impact of each of the seven proposed attributes on the efficacy of the proposed classification method, each attribute was systematically excluded from the classification algorithm, and the effects on the precision metric of the method were assessed on each of the four classification categories (Fig. 10). For example, -ELEV means the voxel s elevation was excluded from the classification algorithm. The results show that in the case of ground classification (Fig. 10a), when the attribute elevation was excluded (-ELEV) from the algorithm the average precision across the 4 tests decreased from 91% to 62%. In contrast, the roof classification only fell to 82% (Fig. 10b) with the removal of the same attribute. For the walls (Fig. 10c) and vegetation (Fig. 10d), excluding the attribute (CLUS) had a more detrimental effect. This small experiment shows the importance of all the proposed attributes. Here they are only shown for precision, but the recall results were very similar.

Table 2
Distribution of voxel categories of the Dublin study area and its folds.

Class	Total Study Area	Fold 1	Fold 2	Fold 3	Fold 4
1. Ground	14.25%	3.07%	3.28%	4.29%	3.62%
2. Roof	31.41%	7.76%	8.90%	7.38%	7.37%
3. Wall	44.67%	12.01%	11.54%	10.08%	11.04%
4. Vegetation	9.67%	2.16%	1.28%	3.25%	2.97%

Table 3
Testing Design and Results.

	Validation Fold	Training Folds	Precision (av.)	Recall (av.)	F1 score (av.)
Test 1	1	2,3,4	0.92	0.92	0.92
Test 2	2	1,3,4	0.89	0.90	0.90
Test 3	3	1,2,4	0.93	0.91	0.92
Test 4	4	1,2,3	0.95	0.92	0.93
Overall average			0.92	0.91	0.92

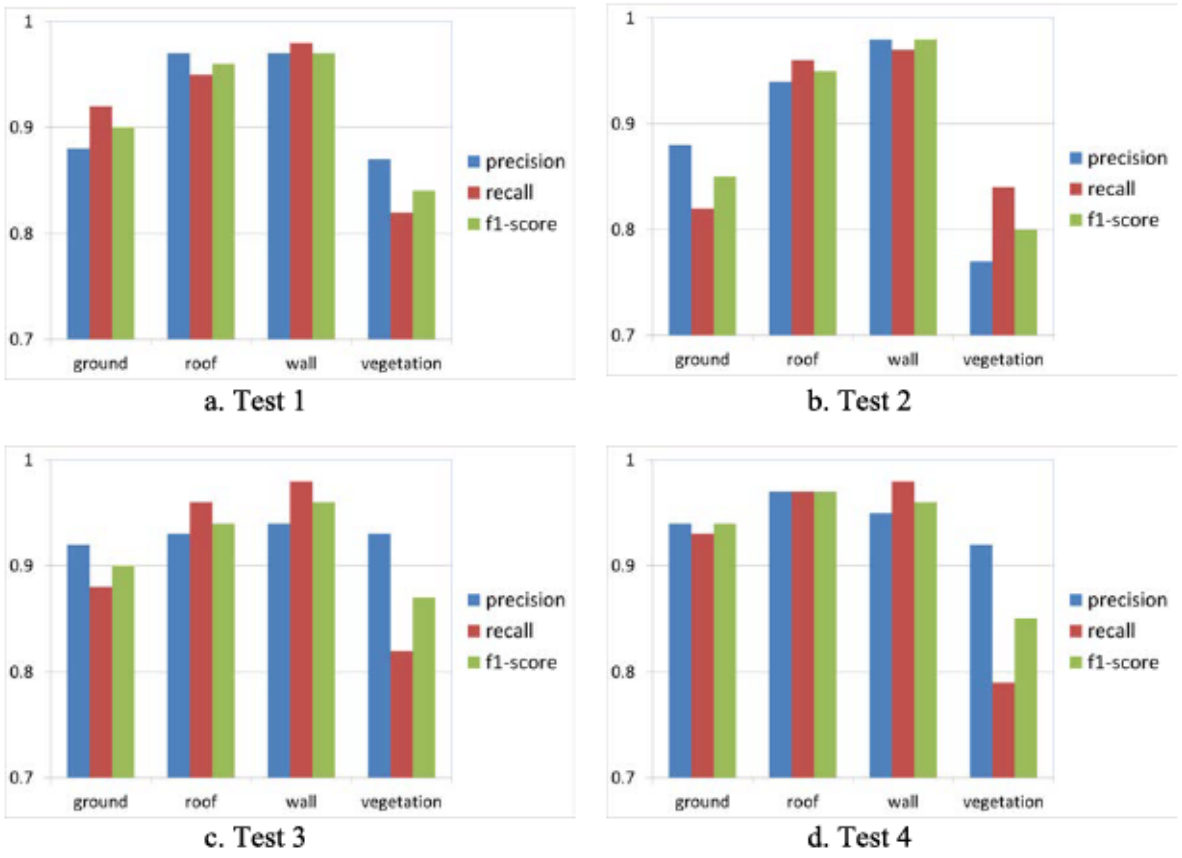


Fig. 8. Classification results of the 4 tests.

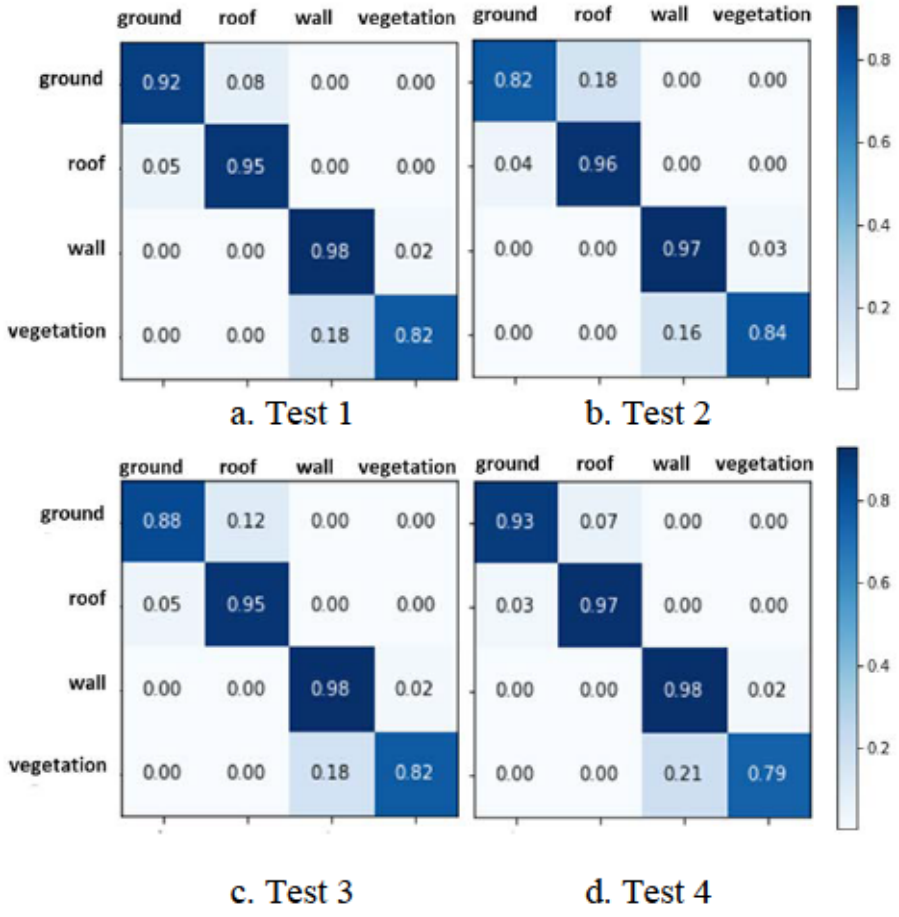


Fig. 9. Confusion matrix results of the 4 tests.

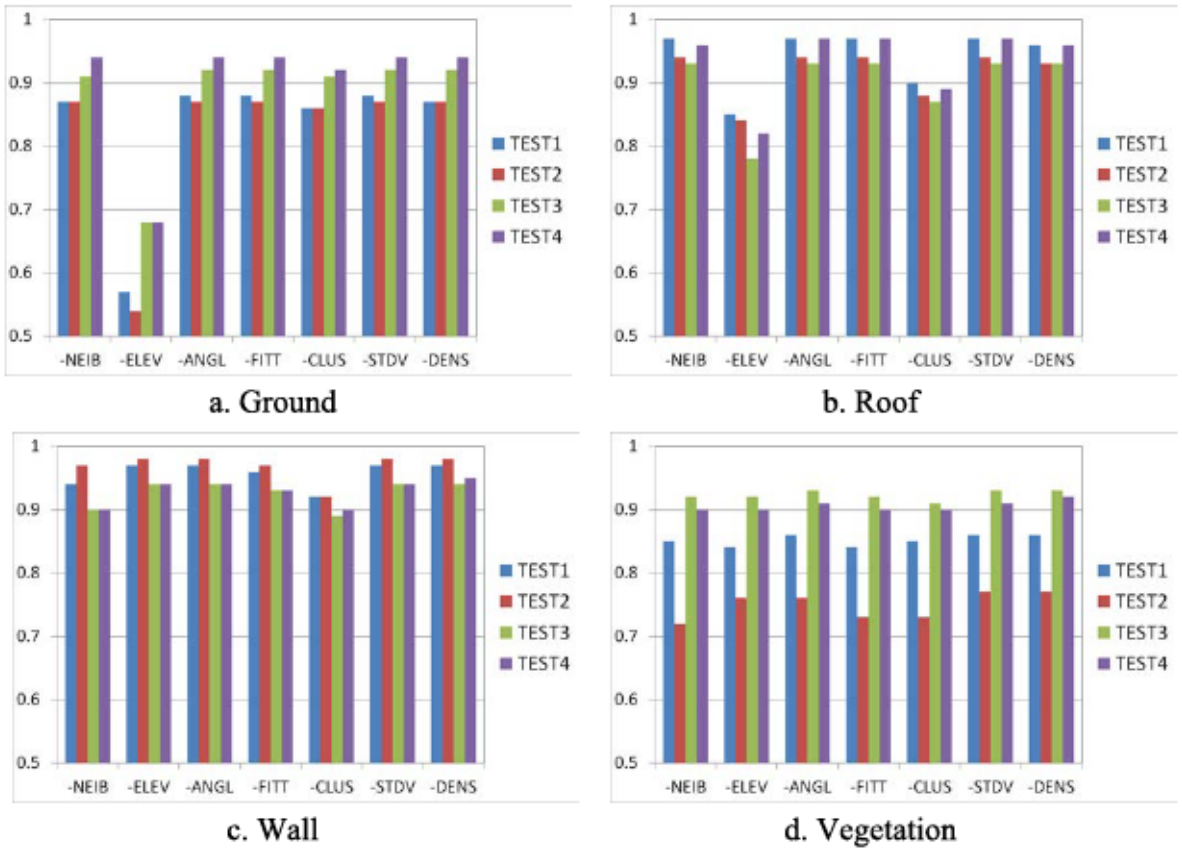


Fig. 10. Effect of excluding attributes from the algorithm on precision results.

5. Discussion

The proposed approach offers notable advancements in the field of automatic aerial LiDAR point classification including the classification of 8 GB of data without use of extensive computational capabilities. The proposed approach was able to obtain a high classification results (Fig. 11) by processing just 3D point coordinates, as can be seen through visual inspection. Scanner-specific, LiDAR attributes (e.g. timestamp, an intensity measurement, Red-Green-Blue color indicators) were discarded to make the PCVC approach applicable to any point cloud of sufficient density.

The proposed PCVC approach classified all 4 categories at relatively high F1 levels, with much of the misclassification involving walls, in large part due to the low data resolution on vertical surfaces (approximately 1 vertical point per every 10 on horizontal surfaces) [Hinks et al. 2009, Stanley and Laefer, 2021]. Some confusion also existed between the ground and roof classifications (from 3% to 18%), as the ELEV term is relative, since there is not a benchmarked ground plane. A comparison

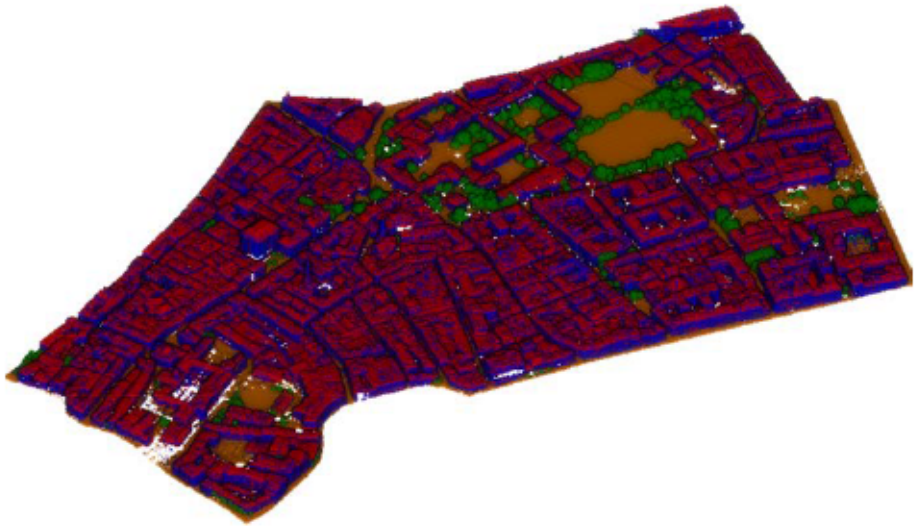


Fig. 11. Case study 1 Dublin, Ireland visualization results for the four classes after application of the PCVC algorithm (brown = ground, red = roofs, blue = walls, green = vegetation) using 3 million points each with a 10-attribute vector versus the 300 million original points. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

of the PCVC approach to previous results on the same study area (Aljumaily et al. 2021) demonstrated a consistent superiority with an average improvement in percentage points of 8.20 for precision, 2.26 for Recall, and 5.06 of the F1-score (Table 4).

To further validate the quality of the classification method, the approach was applied to a second publicly-accessible LiDAR dataset. This dataset is for a portion of Brooklyn, New York and was captured in 2019 (Laefer and Vo, 2020). The new dataset is approximately 50% denser than the Dublin data set. From New York data, a single tile (T_978500_174500) of 150 m × 150 m was used, which consisted of 13,912,692 points. These points were mapped to 89,290 voxels resulting in an average of 156 points per voxel (versus the 116 for the Dublin dataset). As was done for Dublin dataset, a K-Fold Cross Validation technique was used. Notably, in this study, transfer learning was applied between the study areas. The distribution between categories from the four equisized folds of the Brooklyn point cloud is shown in Table 5.

The results of the classification of this second dataset are shown in Figs. 12 and 13. Fig. 13 illustrates the second case study (Brooklyn, New York City) visualization results for the four classes after application of the PCVC algorithm (brown = ground, red = roofs, blue = walls, green = vegetation) using 1 million points versus the 100 million original points. In this case, the lowest classification for any category was 91%, which was a major improvement for the vegetation (previously only 79%-84%). The ground classification improved slightly and the buildings and walls decreased a little. While the misclassification of vegetation for walls decreased 6% (from 16% to 21% in Case Study 1), it was the highest error level of all of the categories. Of particular note is the prominence of vehicles in this second case study. These were classified as part of the ground as part of the training process. As the scanner in this data set was faster, cars were more prominent. Fig. 13 identifies a few areas of misclassification, as denoted by the white circles. For example, circles 1 and 2 show some vegetation voxels that were misclassified as

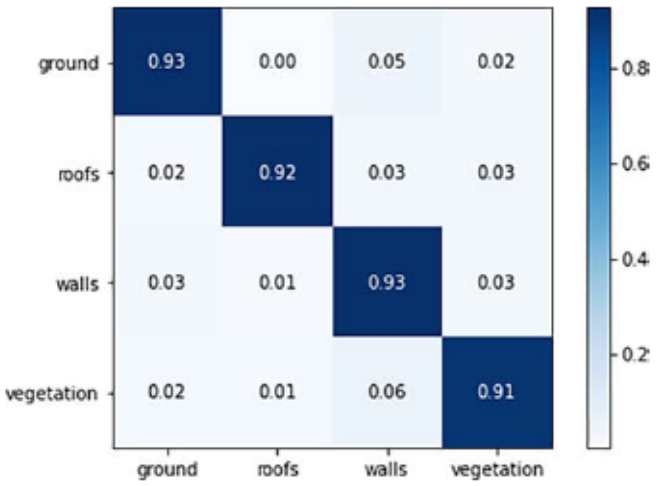


Fig. 12. Confusion matrix of classification results of a second dataset.

ground. Circle 3 shows some of the wall voxels that were misclassified as roof voxels, and in circle 4 are roof voxels that were misclassified as vegetation voxels (see Fig. 14 for a visualization of examples of this type of data configuration).

Fig. 14 shows an example of data where the approach confused the walls and the vegetation. The voxel on the left of the figure is a vegetation voxel, however, the PCVC approach classified it as a wall voxel. In contrast, the voxel n the right of the figure is a wall voxel, but the approach classified it as a vegetation voxel. Reasons for such confusion can be attributed to the complexity of the manual extraction and because vegetation voxels do not have clear distribution patterns. A limitation of the proposed PCVC is that objects smaller than or equal to 1 m^3 are classified as noise. However, in this data set, these represented only 0.01% of the total voxels. Thus, the impact of this limitation is quite modest

Critical to the assessment of the worth of the PCVC algorithm is its scalability. To demonstrate this, the required execution time is shown in Fig. 15. The blue line represents the execution time needed for voxels division, the green line is for calculating the descriptive attributes, and the red line shows the time needed for the RF algorithm. For example, to segment 10,000,000 voxels the MapReduce step needed approximately 25 s, a further 80 s was required for calculating the descriptive attributes, and a additional 160 s for the RF algorithm. This was tested at 10 million, 50 million, 100 million, 150 million, and 200 million points.

While these results clearly show the potential effectiveness of PCVC, several parameters have yet to be fully explored. While grid-size may be further optimized, the foremost topic should be establishing a minimum viable density and an optimum density. As aerial LiDAR data sets get even denser, there may ultimately be cause for establishing a maximum useful density.

Finally, the proposed PCVC approach is considered with respect related work such as Wang et al. (2015), Yun and Sim (2016), Guan et al. (2016), Xu et al. (2019), and Jin et al. (2022). In Guan et al. (2016) the procedure to generate the voxels uses information about the classes it wants to detect. In that case, streetlights and traffic signs are used, which unlike a full urban scene, are entities with a highly distinctive and reliable characteristic structure. Those authors reported 39 min for voxelization and 42 min for class detection, for a dataset with 1,728 million points after the points had been pre-segregated from the ground plane; no time requirement was reported for pre-segregation.

Table 4
Comparison of results compared to previous classification of the study area (*In the previous study the roofs and walls were not considered separately, so they are reported here as a merged category for ease of comparison).

		Precision	Recall	F1 score
PCVC approach	Ground	90.50	88.75	89.75
	Building*	95.25	96.00	95.50
	Vegetation	87.25	81.75	84.00
Average		91.00	88.83	89.75
Previous study (Aljumaily et al., 2021)	Ground	88.74	92.86	90.75
	Building	81.85	86.64	84.18
	Vegetation	77.81	80.23	79.00
Average		82.80	86.58	84.69

Table 5
Description of the New York study area and its folds.

Class	Voxels	Fold 1	Fold 2	Fold 3	Fold 4
1. Ground	26.12%	6.92%	4.20%	9.16%	5.84%
2. Roof	9.79%	2.57%	3.24%	1.77%	2.21%
3. Wall	35.77%	8.97%	15.04%	4.45%	7.31%
4. Vegetation	28.33%	10.71%	1.49%	13.11%	3.02%

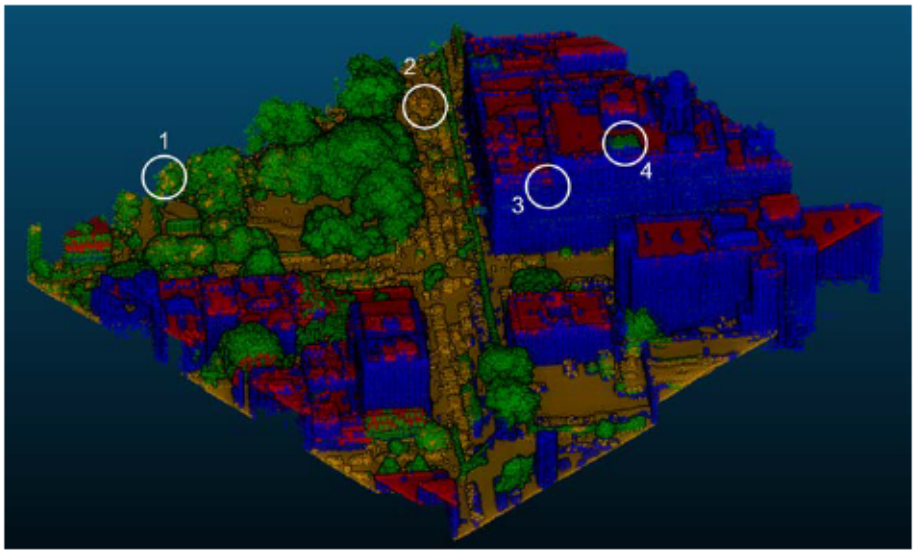


Fig. 13. Case study 2 Brooklyn, New York City visualization results for the four classes after application of the PCVC algorithm. White circles are select areas of misclassification.

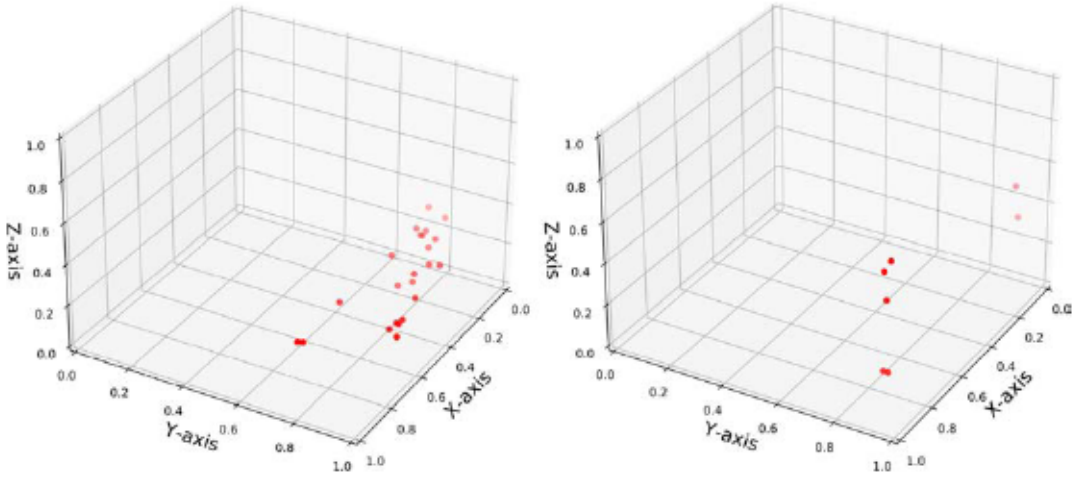


Fig. 14. Vegetation voxel mistaken for wall (left) and wall voxel mistaken for vegetation (right) from case study 1 data.

In similar work, Wang et al. (2015) compared different voxelization methods for detecting streetlights, traffic signs, and cars. The authors noted that a limitation of their work is the inability to detect non-solid objects (i.e. tree canopies and bushes). In the supervoxel approach proposed by Yun and Sim (2016), there is a reliance on integrated imagery. This makes data acquisition and processing more complicated and expensive and precludes night and low-level light data acquisition missions.

In the more recent work by Xu et al. (2019), a supervised semantic labelling algorithm was devised to reduce the training set size by using RF to voxel characterization attributes.. and outperformed PointNet and PointNet++ in all classes except low vegetation. However, LiDAR data sets were from terrestrial scanners so a direct performance comparison with PCVC is not possible, as the aerial data are comparatively sparse and contain major imbalances between the data on vertical and horizontal surfaces, as a function of flight mission geometry. While the disparity can be employed to distinguish road and roof surfaces from other entities, the relatively low data availability for non-horizontal surfaces has proved a largely insurmountable issue in adaptation of terrestrial point cloud processing techniques for use with aerial point clouds. Most recently, Jin et al. (2022), employed aerial data for urban scene classification but their hybrid data approach which relies on a joint LiDAR-hyperspectral imagery data, is extremely cumbersome to process and relatively expensive to obtain. In contrast, in the proposed PCVC, RF is used as means for choosing a set of features that adequately discriminate. As shown in the results, the proposed attributes are easy to derive from the points themselves within each voxel, discriminate well, and were able to reduce the datapoints in the classification task by approximately 99%.

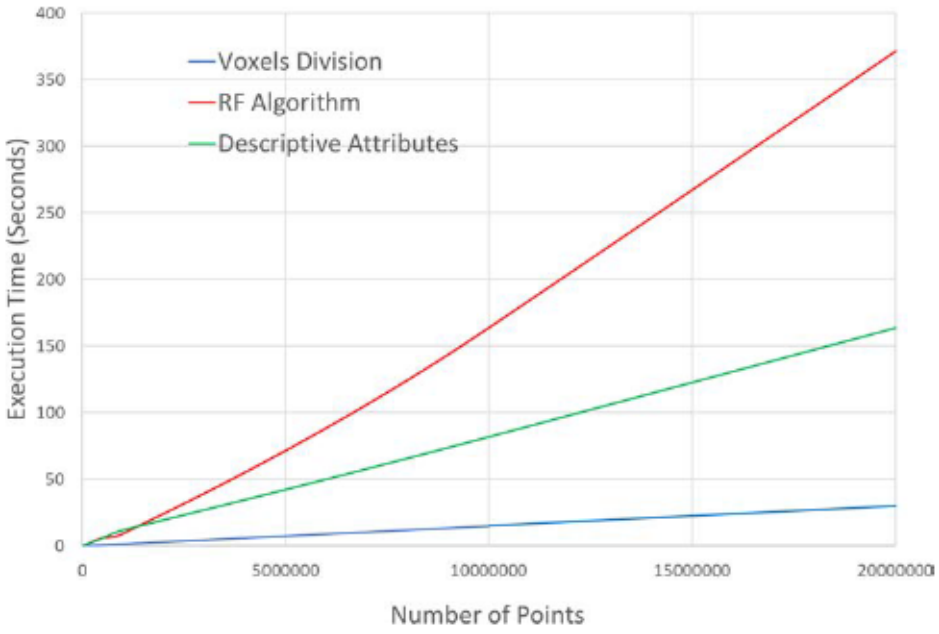


Fig. 15. Execution time needed for the proposed PCVC.

6. Conclusions

Scalability poses a major impediment for most object classification and extraction approaches. To address this problem, the proposed PCVC approach offers a fully automatic and significantly more scalable approach to LiDAR classification by reducing approximately 99% of the processed points when applying the RF algorithm. This reduction enable processing to occur on a single machine, instead of having to adopt a distributed computing solution. The PCVC approach was tested on a study area of more than 300 million points over approximately 1 km^2 and typically reached classification accuracy in excess of more than 94%.

The PCVC approach creates 3D grid-based subspaces that are independent on sensor type and data format. This implies greater scalability addressing the problem raised by Bassier et al., (2019) where the data collection device was enhanced to improve the classification. PCVC overcomes the necessity for manual parameter selection by using mathematical and robust algorithms to classify the subspaces. While manystate of the art approaches [e.g. (Park and Guldmann (2019); Azadbakht et al. (2016)], employ a employ mixed point geometry, context geometry (for each subspace), and global attributes to create a model able to classify different classes, PCVC only uses point geometry. While the importance of all of the selected attributes was demonstrated, the ELEV feature dominated the classification proces, especially for classifying Ground, with with the absence of that feature causing a 40% reduction in accuracy. To enhance the PCVC procedure, future work is needed on data density sensitivity, voxel size optimization, and the potential expansion of categories for classification.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was funded by the National Science Foundation award 1940145.

References

Ahmed, O.S., Franklin, S.E., Wulder, M.A., White, J.C., 2015. Characterizing stand-level forest canopy cover and height using Landsat time series, samples of airborne LiDAR, and the Random Forest algorithm. ISPRS J. of Photogramm. and Remote Sens. 101, 89–101.

Aljumaily, H., Laefer, D.F., Cuadra, D., 2015. Big-Data Approach for Three-Dimensional Building Extraction from Aerial Laser Scanning. J. Comput. Civ. Eng. 30 (3), 04015049.

Aljumaily, H., Laefer, D.F., Cuadra, D., 2017. Urban Point Cloud Mining Based on Density Clustering and MapReduce. J. of Comput. in Civil Eng. 31 (5), 04017021.

Aljumaily, H., Laefer, D.F., Cuadra, D., 2019. Integration of lidar data and GIS data for point cloud semantic enrichment at the point level. Photogramm. Eng. and Remote Sens. 85 (1), 29–42.

Aljumaily, H., Laefer, D.F., Cuadra, D., Velasco, M., 2021. Voxel Change: Big Data-Based Change Detection for Aerial Urban LiDAR of Unequal Densities. J. of Surveying Eng. 147 (4), 04021023.

Awrangjeb, M., Zhang, C., Fraser, C.S., 2013. Automatic extraction of building roofs using LiDAR data and multispectral imagery. ISPRS J. of Photogramm. and Remote Sens. 83, 1–18.

Azadbakht, M., Fraser, C.S., Khoshelham, K., 2016. Improved urban scene classification using full-waveform lidar. Photogramm. Eng. and Remote Sens. 82 (12), 973–980.

Bassier, M., Van Genechten, B., Vergauwen, M., 2019. Classification of sensor independent point cloud data of building objects using random forests. J. of Building Eng. 21, 468–477. ISSN 2352–7102.

Belgiu, M., Drăguţ, L., 2016. Random forest in remote sensing: A review of applications and future directions. ISPRS J. of Photogramm. and Remote Sens. 114, 24–31.

Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32.

Chakrawarty, L., Gupta, P., 2014. Applying SR-Tree technique in DBSCAN clustering algorithm. Int. J. of Application or Innovation in Eng. and Management. 3 (1), 207–210.

Chehata, N., Guo, L., Mallet, C., 2009. Airborne lidar feature selection for urban classification using random forests. Int. Arch. Photogramm. Remote Sens. XXXVIII-3/W8, 207–212.

CloudCompare, 2015. 3D point cloud and mesh processing software. http://www.cloudcompare.org/doc/qCC/CloudCompare%20v2.6.1%20-%20User%20manual.pdf.

Ester, M., Kriegel, H.P., Sander, J., Xu, X., 1996, August. A density-based algorithm for discovering clusters in large spatial databases with noise. In kdd, Vol. 96, No. 34, pp. 226-231.

Farnaaz, N., Jabbar, M.A., 2016. Random forest modeling for network intrusion detection system. Procedia Comp. Sci. 89, 213–217.

Fischler, M.A., Bolles, R.C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24 (6), 381–395.

Guan, H., Yu, Y., Li, J., Liu, P., 2016. Pole-Like Road Object Detection in Mobile LiDAR Data via Supervoxel and Bag-of-contextual-visual-words Representation. IEEE Geosci. Remote Sens. Lett. 13 (4), 520 524.
Guo, B., Huang, X.F., Zhang, F., Sohn, G.H., 2015. Classification of airborne laser scanning data using jointboost. ISPRS-J. Photogramm. Remote Sens. 100, 71 83.
Hinks, T., Carr, H., Laefer, D.F., 2009. Flight optimization algorithms for aerial LiDAR capture for urban infrastructure model generation. J. of Comp. in Civil Eng. 23 (6), 330 339.
X, Jin, Y, Gu, T, Liu, W, Xie, Supervoxel-Based Intrinsic Scene Properties From Hyperspectral Images and LiDAR, in IEEE Transactions on Geoscience and Remote Sensing, vol. 60, pp. 1-13, 2022, Art no. 5510613 10.1109/TGRS.2021.3096839.
Khan, R., Hanbury, A., Stoettinger, J., 2010. Skin detection: a random forest approach. IEEE Int. Conference on Image Processing. 4613 4616.
D.F, Laefer, A.V, Vo, 2020. 2019 LiDAR Data Collection for Sunset Park. NYU Spatial Data Repository, Brooklyn, NY doi.org/10.17609/4cpx-2h33/. Available at http://hdl.handle.net/2451/60458.
Li, M., Sun, C., 2018. Refinement of LiDAR point clouds using a super voxel based approach. ISPRS J. of Photogramm. and Remote Sens. 143, 213 221.
Lin, Y., Vosselman, G., Cao, Y., Yang, M.Y., 2020. Active and incremental learning for semantic ALS point cloud segmentation. ISPRS J. of Photogramm. and Remote Sens. 169, 73 92.
Liu, W., Sun, J., Li, W., Hu, T., Wang, P., 2019. Deep learning on point clouds and its application: a survey. Sensors (Basel, Switzerland) 19 (19), 4188. https://doi.org/10.3390/s19194188.
Naidoo, L., Cho, M.A., Mathieu, R., Asner, G., 2012. Classification of savanna tree species, in the Greater Kruger National Park region, by integrating hyperspectral and LiDAR data in a Random Forest data mining environment. ISPRS J. of Photogramm. and Remote Sens. 69, 167 179.
Ni, H., Lin, X., Zhang, J., 2017. Classification of ALS point cloud with improved point cloud segmentation and random forests. Remote Sens. 9 (3), 288. https://doi.org/10.3390/rs9030288.
Nikoohemat, S., Peter, M., Oude Elberink, S., Vosselman, G., 2017. Exploiting indoor mobile laser scanner trajectories for semantic interpretation of point clouds. ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. IV-2/W4, 355 362.
T. M, Oshiro, P. S, Perez, J. A, Baranauskas, How many trees in a random forest?, in: International workshop on machine learning and data mining in pattern recognition. Springer, Berlin, Heidelberg 2012, pp. 154-168.
Park, Y., Guldmann, J.M., 2019. Creating 3D city models with building footprints and LIDAR point cloud classification: a machine learning approach. Comput. Environ. Urban Syst. 75, 76 89. ISSN 0198 9715.
Previtali, M., Scaioni, M., Barazzetti, L., Brumana, R., 2014. A flexible methodology for outdoor/indoor building reconstruction from occluded point clouds. ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. II-3, 119 126.
Python Core Team, 2019. Python: A dynamic, open source programming language. Python Software Foundation. https://www.python.org/.
C.R, Qi, H, Su, K, Mo, L.J, Guibas, PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation 2016, arXiv preprint arXiv:1612.00593.
A, Ramesh, I, Petousis, O, Davis, C, Heiser, Systems and methods for providing resource analysis for autonomous mobility on demand. U.S. Patent Application 2019 No. 16/208,418.
Soilan, M., Riveiro, B., Balado, J., Arias, P., 2020. Comparison of heuristic and deep learning-based methods for ground classification from aerial point clouds. Int. J. of Digital Earth 13 (10), 1115 1134. https://doi.org/10.1080/17538947.2019.1663948.
M.H, Stanley, D.F, Laefer, Metrics for aerial, urban LiDAR point clouds. ISPRS J. of Photogramm. and Remote Sens. 2021 175, 268-281 10.1016/j.isprsjprs.2021.01.010.
L, Truong-Hong, D. F, Laefer, T, Hinks, H, Carr, Combining an angle Criterion with voxelization and the flying voxel method in reconstructing building models from LiDAR data. Computer-Aided Civil and Infrastructure Eng. 28 2 2013 112-129 10.1111/j.1467-8667.2012.00761.x.
U.S. Geological Survey (USGS), 2020. https://www.usgs.gov/.
Vosselman, G., Coenen, M., Rottensteiner, F., 2017. Contextual segment-based classification of airborne laser scanner data. ISPRS J. of Photogramm. and Remote Sens. 128, 354 371. ISSN 0924 2716.
Vu, A.V., Truong-Hong, L., Laefer, D.F., Bertolotto, M., 2015. Octree-based region growing for point cloud segmentation. ISPRS J. of Photogramm. and Remote Sens. 104, 88 100.
Wang, C., Shu, Q., Wang, X., Guo, B., Liu, P., Li, Q., 2019. A random forest classifier based on pixel comparison features for urban LiDAR data. ISPRS J. of Photogramm. and Remote Sens. 148, 75 86.
Wang, H.C., Wang, H., Luo, P., Li, Y.C., Li, J., April 2015. 3-D point cloud object detection based on supervoxel neighborhood with hough forest framework. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 8 (4), 1570 1581. https://doi.org/10.1109/JSTARS.2015.2394803.
Weinmann, M., Jutzi, B., Hinz, S., Mallet, C., 2015. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. ISPRS J. Photogramm. Remote Sens. 105, 286 304.
B, Xiang, J, Yao, X, Lu, L, Li, R, Xie, Segmentation-based classification for 3D urban point clouds, in: 2016 IEEE International Conference on Information and Automation (ICIA), Ningbo 2016, pp. 172-177.
Xu, Y., et al., 2019. Classification of LiDAR point clouds using supervoxel-based detrended feature and perception-weighted graphical model. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 13, 72 88. https://doi.org/10.1109/JSTARS.2019.2951293.
Xu, Y., Stilla, U., 2021. Toward building and civil infrastructure reconstruction from point clouds: a review on data and key techniques. IEEE J. of Selected Topics in Appl. Earth Observations and Remote Sens. 14, 2857 2885. https://doi.org/10.1109/JSTARS.2021.3060568.
Xu, S., Vosselman, G., Elberink, S.O., 2014. Multiple-entity based classification of airborne laser scanning data in urban areas. ISPRS J. of Photogramm. and Remote Sens. 88, 1 15.
Yang, B.S., Dong, Z., Zhao, G., Dai, W.X., 2015. Hierarchical extraction of urban objects from mobile laser scanning data. ISPRS J. Photogramm. Remote Sens. 99, 45 57.
Yastikli, N., Cetin, Z., 2016. Classification of LiDAR data with point based classification methods. Int. Arch. Photogr. Remote Sens. Spat. Inf. Sci. 41, 441 445.
Yun, J., Sim, J., 2016. Supervoxel-based saliency detection for large-scale colored 3D point clouds. IEEE Int. Conference on Image Processing (ICIP) 4062 4066. https://doi.org/10.1109/ICIP.2016.7533123.
Zhang, J., Hu, X., Hengming, D., Qu, S., 2020. DEM Extraction from LiDAR point clouds in forest areas via graph convolution network. Remote Sens. 12, 178. https://doi.org/10.3390/rs12010178.
S. M, Zolanvari, S, Ruano, A, Rana, A, Cummins, R. E, da Silva, M, Rahbar, A, Molic, DublinCity: Annotated LiDAR Point Cloud and its Applications 2019.