Final version as published in the ASCE Journal of Computing in Civil Engineering available at
https://dx.doi.org/10.1061/(ASCE)CP.1943-5487.0000674

Urban Point Cloud Mining Based on Density Clustering and MapReduce

Harith Aljumaily[*], Debra F. Laefer[**], and Dolores Cuadra[*]

[*] Department Computer Science and Engineering
Carlos III University of Madrid
Av. Universidad 30 – 28911 – Madrid, Spain
{haljumai, dcuadra}@inf.uc3m.es

[**] School of Civil, Structural and Environmental Engineering;
U3D Printing Hub & Earth Institute
University College Dublin
Newstead G25, Belfield, Dublin 4, Ireland
debra.laefer@ucd.ie

ABSTRACT: This paper proposes an approach to classify, localize, and extract automatically urban objects such as buildings and the ground surface from a digital surface model created from aerial laser scanning data. To achieve that, the approach involves three steps: 1) dividing the original data into smaller, more manageable pieces using a method based on MapReduce gridding for subspace partitioning; 2) applying the DBSCAN algorithm to identify interesting subspaces depending on point density; and 3) grouping of identified subspace to form potential objects. Validation of the method was achieved using an architecturally dense and complex portion of Dublin, Ireland. The best results were achieved with a 1 m^3 sized clustering cube, for which the number of classified clusters equaled that which was derived manually and that amongst those there the following scores: correctness = 84.91%, completeness = 84.39%, and quality = 84.65%.

KEYWORDS: Building Extraction, MapReduce, Big Data, LiDAR, DBSCAN algorithm, Clustering Classification approaches

INTRODUCTION

Urban Modelling (UM) benefits from spatial data mining to detect, localize, and extract geographic objects such as the ground surface, vegetative regions, and manmade objects. Such datasets may come from satellite, Light Detection And Ranging (LiDAR), environmental sensors, and even social networks. Traditionally such UM datasets have been stored and visualized in a Geographic Information System (GIS) or a spatial database to be used for civil, political, or commercial

applications. However, the increasing density of such data now challenges its most basic functionality and usefulness. A key challenges includes how to achieve data analysis in a computationally efficient way in huge datasets without overwhelming the computational infrastructure. Normally, a small Digital Surface Model (DSM) of a limited area derived from LiDAR point cloud data would consist of several million to a few billion three-dimensional (3D) points. Each point is typically affiliated with 3D coordinates, a timestamp, an intensity measurement, and possibly Red-Green-Blue colour indicators, if there is a co-registered image. Making these points compatible with a Spatial Data Model (i.e. generating a geo-Identification Key and Spatial Objects for each point) requires significantly more storage than that needed to host the original dataset. Given the rapid trajectory of LiDAR density growing at nearly an order of magnitude per decade (see Vu et al. 2016), traditional LiDAR storage solutions will only increasingly struggle to support the rapidly escalating number of LiDAR users and the ever-expanding types of queries. For these reasons, Big Data platforms can offer a logical and useful choice to store and analyze large-scale, urban, spatial data generated from LiDAR point clouds.

A related issue is object identification within large data sets. Of growing popularity are clustering based approaches to group similar data objects (e.g., Fu et al. 2014) for storage and querying (e.g., Kurasova et al. 2014). While there are many well-known algorithms to find data clusters depending on the distance metrics between objects or points, Kailing et al. (2004) noted that these algorithms often fail to detect meaningful clusters in datasets with large differences in densities and/or in the presence of high-dimensional, real-world data sets. More dimensions mean more distance between points, which compromises efficiency. Notably, many clustering approaches are already quite memory intensive. Therefore, their scalability is highly uncertain. Consequently, implementing a clustering approach in a Big Data context holds the promise of addressing these problems directly and offers the potential for unprecedented efficiency. To this end, this paper introduces a new, fully automatic approach for cluster-based data mining of LiDAR data with no reliance upon pre-processing and usage of only the 3D coordinate information. The approach takes raw 3D points of a given LiDAR-based DSM and converts them into sets of clusters, where each cluster is a set of high density points. A cluster represents an object such as a building or a ground surface. The remainder of this work is organized as follows: Section 2 reviews the peer-reviewed literature in the field of clustering mining and big data; Section 3 describes the approach in details; Section 4 presents series of validation experiments; and Section 5 formulates general conclusions and identifies areas for future research.

RELATED WORKS

Data mining for the purpose of building segmentation, extraction, and reconstruction is a well-established topic within the geomatics community. The general approaches have been either procedural in nature and require predefined geometries or libraries or they have been data driven. These often rely upon voxelization (e.g., Vo et al. 2015) or k-nearest neighbour approaches (e.g., Truong-Hong et al. 2013). The techniques have often been cross-applied to laser scanning (terrestrial, mobile, and aerial), photo-based imagery, and a combination of the two (e.g., the Dempster-Shafer theory for data fusion by Rottensteiner et al. 2004; three-dimensional roof extraction using laser scanning data and multispectral orthoimagery work by Awrangjeb et al. 2013; building boundary detection with laser scanning and optical imagery by Li et al. 2013; and raster and point cloud based GIS analysis by Jochem et al. 2012).

Clustering approaches have been widely used for various proposes such as data mining, image analysis, and machine learning. Generally, these approaches are used to find regions in a predefined space (Chakrawarty et al. 2014). In the context of urban data mining, these approaches are mainly used to extract a set of patterns, points, or objects from the data. Many clustering algorithms have been published since the early introduction of K-NN algorithm (Truong-Hong et al. 2013) and the K-means algorithm (Jain 2010) in the decade of the 1950s. K-NN is a partitioning approach based on a classification algorithm that aims to split a space into k clusters. A K-means is a partitioning based clustering algorithm that is used to cluster N objects into K clusters depending upon the distance between the centres of the clusters. More recently, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm was introduced by Ester et al. (1996) to find arbitrarily shaped clusters, handle noise, and address data of any type in clustering. The main difference between these algorithms is that K-NN and K-means are considered partitioning algorithms, and they are relatively sensitive to the outliers, which means having outliers would reduce their accuracy (Chen et. al. 2006). On the other hand, since DBSCAN is a clustering algorithm based on density, it is realtive insensitive to the outliers (Xu and Tian, 2015). This means that DBSCAN is robust towards outlier detection (Noise) and, thus, selected for implementation herein.

The basic idea of DBSCAN is that a cluster is formed around a core point, if and only if, the neighbourhood of a given radius has a minimum number of points. Since its introduction in 1996, DBSCAN has been used extensively and continuously in this field. Recent examples include the work by Lee et al. (2014), who proposed a framework based on DBSCAN as a default clustering algorithm to extract associated points-of-interest patterns from geo-tagged photos. In related work, Zhou et al. (2015) employed DBSCAN to detect the geographic locations of tourism destinations from geo-tagged digital photos, while in Wang et al. (2013), DBSCAN was used for automated 3D

buildings reconstruction from LiDAR Data. The authors first detected the building outlines and then reconstructed the models.

Although this algorithm and many others are well-established and extensively used with success on limited datasets, should they be applied indiscriminately to a large dataset, their computational expense would likely overwhelm the process. In contrast a subspace clustering technique holds the potential to improve the data mining speed and the ability to detect robustly clusters of interest. Such an approach can reduce retrieval time by minimizing the number of records accessed (Parsons et al. 2004). In this case, the whole space of the problem is divided into smaller subspaces, and each subspace contains a piece of a cluster. Subspaces with density above of a defined threshold are selected as potential members of a cluster. In order to divide a whole space into smaller subspaces and to find potential clusters, grid-based clustering methods can be applied (e.g., Chang et al. 2002 and 2005; Parsons et al 2004). Clustering methods have a series of common steps (Aggarwal et al. 2013) starting with creating the grid structure with a finite number of subspaces, proceeding to calculating the density for each subspace, then sorting the subspaces according to their densities, followed by identifying the cluster centres, and finally by traversing of neighbour subspaces. As an extension of this, Darong et al. (2012) proposed a combination of a grid-based partition technique and the DBSCAN algorithm. Those experimental results showed that this combination improved not only the segment separation between clusters and noise but proved also to be more robust.

While these various studies have produced important results for building extraction, few have considered the trajectory of the rapidly escalating size of point cloud datasets with respect to their spatial extent and their density. The current generation of data readily attains 50 pt/m^2 with data sets of 225 pt/m^2 being publicly available (e.g., UCD Digital Library, (2007)). Thus, a spatial Big Data context appears as an inevitable requirement. The work presented in Zhang et al. (2009) described how spatial queries could be adopted and expressed in a MapReduce framework, which is the key of the Big Data processing. A MapReduce framework is a software model used to support parallel computing of huge sets of data and consists of two functions Map and Reduce, which operate using key-value data types. The function 'Map' processes the original data into key/value pairs, and the function 'Reduce' takes these pairs and merges them in a way that all values corresponding to a specific key are combined into a single set. Zhou et al. (2015) using DBSCAN on a Hadoop distributed system demonstrated the ability to support a scalable geoprocessing workflow and expedite geospatial problem solving, as previously predicted by Fu et al. (2014). The improved scalability stems from the framework’s division of the input dataset into smaller parts and its subsequent outward distribution of them to nodes for parallel processing. In a generic sense, Wang et al. (2010) demonstrated experimentally that more nodes in a cluster significantly improve

the execution time of the MapReduce processing. Recently, a Big Data approach for buildings extraction from a DSM was introduced by Aljumaily et al. (2015). The approach first employed a MapReduce process where neighboring points are mapped into subspaces as cubes. Next, a non-MapReduce algorithm was used to remove trees and other obstructions. Finally all adjacent cubes belonging to the same object were extracted based on defining an object as a set of adjacent cubes that belong to one or more adjacent buildings.

CLUSTERING APPROACH

The goal of the work presented herein is to perform in a Big Data context clustering classification on raw LiDAR data without reliance on pre-processing. The main objectives of the clustering classification are to (A) remove vegetation and other obstructions from the DSM and (B) detect and localize outlines of urban objects such as buildings and the ground surface. The proposed approach involves three steps involving (1) MapReduce grid-based partitioning; (2) dense subspace detection; and (3) object formation. These steps are described in detail in the following subsections.

As part of this work, a $1km^2$ study area in the centre of Dublin Ireland was used. A total of ~225 million points from aerial laser scanning were acquired in the winter of 2007 for a dense urban area of Dublin, Ireland. The data were acquired by contractors using a FLI-MAP 2 system. The system operated at a scan angle of 60 degrees, with an angular spacing of 60/1000 degrees between pulses. While the FLI-MAP 2 system can provide spectral data in the form of intensity and colour, the colour data was not collected. The flying altitude varied between ~380-480m, with an average value of ~400m. Total 44 flight strips were acquired and 2823 flight path points were recorded, providing instantaneous aircraft position over time (for more information see UCD Digital Library, (2007)).

Step-1: A MapReduce Grid-Based Method for Subspace Partitioning

Since, grid-based methods for subspaces partitioning have the great advantage of reducing significantly the time complexity, especially for high dimensional datasets (Aggarwal et al. 2013), the basic idea presented herein is to exploit the coordinates (x, y, z) of each point in the point cloud and then to map these points into smaller subspaces (i.e. cubes). Herein, a cube is considered to belong only to one object. Consequently, if two neighbouring points belong to the same cube, then they belong to the same object. Similarly, two neighbouring cubes are assumed to belong to the same cluster or object, if they have similar internal point distributions, as will be describe subsequently.

As such, the dimension (d) of these cubes plays an important role in the final results of the classification. This is to say, if d is too large, a cube may contain both vegetation and parts of a

building, or two objects may share the same cube. On the other hand, if d is too small, an excessive number of cubes will result, which may be unnecessarily time consuming. Based on a preliminary empirical study, the parameter d was initially selected as 1.0m, because the resulting volume is smaller than most urban objects but relatively insignificant compared to the entire volume of the whole digital surface model, which was 3,248,520 m^3. To test the sensitivity of the value selected for d, three values (0.5m, 1.0m, and 2.0m) were selected for application to the abovementioned dataset, as will by explained in section 4.

To reduce the computational cost of the partitioning process, the MapReduce framework is used as a grid-based method to map the point cloud into cubes (see Figure 1-A). As explained in Section 2, the MapReduce framework is employed to support parallel computing of huge sets of data with the goal to reduce the execution time. While the authors' implementation of the MapReduce framework can be found in detail elsewhere (Aljumaily et al. 2015), it is briefly summarised herein.

Specifically, the point P (x, y, z) is mapped to a cube, which has the identification key equal to KEY=(fix(x), fix(y), fix(z)) (see Figure 1-B). The function fix(v) truncates the value to the greatest integer less than or equal to v. So the Map function receives the point cloud data and issues a list in which each point is mapped to the corresponding cube identification key:

{(KEY_1, $P_{1,1}$), (KEY_1, $P_{1,2}$),…, (KEY_1, $P_{1,N}$),

(KEY_2, $P_{2,1}$), (KEY_1, $P_{2,2}$),…, (KEY_2, $P_{2,M}$), ……

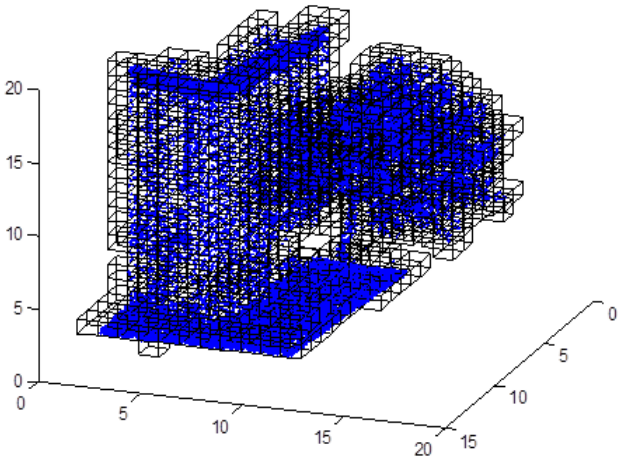
(KEY_L, $P_{L,1}$), (KEY_L, $P_{L,2}$),…, (KEY_L, $P_{L,O}$)}

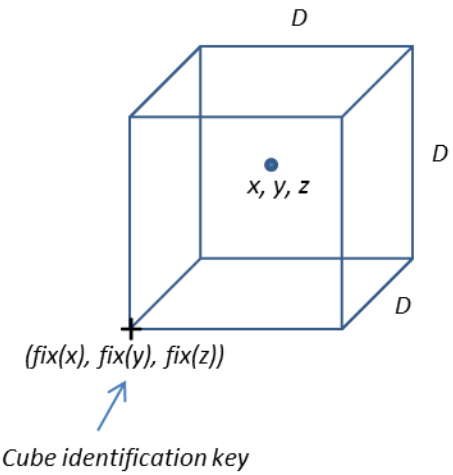
Next, the Reduce function receives the previous list issues a new list, where KEY_i is the cube coordinates, as a unique entry in the list:

{KEY_1, $P_{1,1}$, $P_{1,2}$,…, $P_{1,N}$},

{KEY_2, $P_{2,1}$, $P_{2,2}$,…, $P_{2,M}$}, ….……

{KEY_L, $P_{L,1}$, $P_{L,2}$,…, $P_{L,O}$}





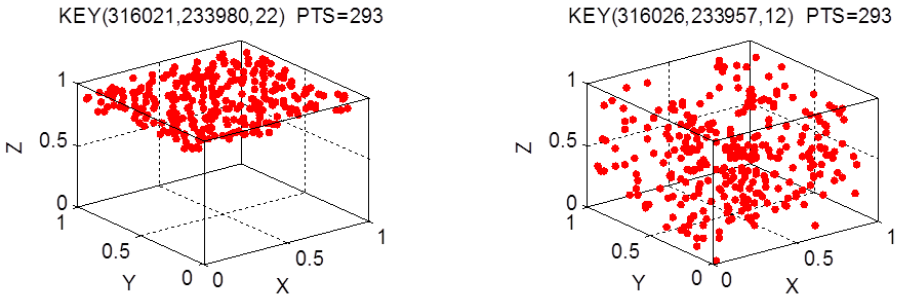
A- 3D objects mapping B- 3D point mapping

Figure 1. Grid-Based Method by MapReduce

At the end of this step, the 3D point cloud is converted into a list where each line in the list forms a cube with its identification key and its points. Once a cube is obtained, it is submitted directly into the next step where a filter-based algorithm is applied to distinguish between cubes that form a cluster and those that do not.

Step-2: Recognizing interesting subspaces

The main supposition of this work is that if a DSM is partitioned into a set of small, grid-based cubes with equal dimensions, two types of cubes can be distinguished within a DSM. The first type is called a dense cube. Each dense cube contains a set of successfully clustered points (Figure 2-A). In that case, the points are grouped together in the cube to form a partially or totally dense (i.e. highly populated) sector within the cube. Of importance is that the approach detects arbitrarily shaped clusters, which mean that cluster formation is independent of the cube's orientation. Normally, a dense cube is involved in forming part of the ground, roads, or buildings (mostly in the form of flat or sloped roofs).



A- Dense cube B- Sparse cube

Figure 2. dense cube vs. sparse cube.

The second cube type is sparse, where the points within the cube are dispersed and occupy more space (Figure 2-B). Although Cube A and Cube B have the same number of points, their respective distributions are highly distinctive. Typically, a sparse cube forms from vegetation, because of the discontinuous nature of the foliage combined with the laser scanner's ability to

penetrate gaps in the canopies hitting leaves, branches, and portions of the ground (Slatton et al. 2008). For this reason, such results will generate more dispersed and less compact cubes than the first type. This second cube type may also include noise and obstructions such as pedestrians, vehicles, road lighting poles, and so on.

In the work herein, dense cubes are interesting subspaces, because they represent parts of urban objects. For this reason, once the total space is segmented into cubes, then these cubes will be submitted to a filter-based algorithm for classification. As previously noted, because the selected cube volume (i.e. 1 m^3) is small in comparison to the DSM's whole volume, when two neighbouring dense cubes exist, they are assumed to belong to the same cluster. In contrast, sparse cubes are generally treated as noise.

To separate these two cube types, the DBSCAN algorithm (Ester et al. 1996) is used. The approach can efficiently partition 'cluster' and 'noise' into a dataset D of points of k-dimensional space. The algorithm states that the neighbourhood of two points p and q is determined by calculating the distance between the two points dist (p,q). Although the distance can be calculated by any type of distance measure. In this work, the Euclidean distance is used. If the distance dist(p,q) is less than Eps (Eps is the maximum radius threshold to delimit the neighborhood of a point p), then p is considered as a core point, and q is its neighbor. On the other hand, a point q is considered as noise, if the distance between q and any near core points is greater than Eps.

In the following equation (eqn 1), N_{Eps} denotes the point q belongs to the neighbourhood region of the point p:

$$N_{Eps}(p) = \{q \in D \mid dist(p,q) \leq Eps\} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad \text{(eqn 1)}$$

A point p is the core of a cluster, if there exists at least MinPts of points in its neighbourhood region, as shown in (eqn 2):

$$|N_{Eps}(p)| \geq MinPts \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \text{(eqn 2)}$$

Thus, defining the two parameters Eps and MinPts is crucial to the clustering process, because if an overly large Eps is selected, then many points that do not belong to the cluster will be included unintentionally. While if an overly small Eps is selected, then points belonging to the cluster may be unintentionally excluded. The same problem arises with respect to MinPts. If a high MinPts is selected, then a low-density cluster may be included, while a low MinPts would exclude a high-density cluster. As will be explained in the next section, to date only an empirical approach has been deployed for threshold determination of these four parameters.

Now that the main idea of the DBSCAN algorithm has been presented, the specifics of its implementation are presented below. This starts with supposing that a Cube C consists of a set of 3D points, as shown in (eqn 3):

$$\{XYZ\} = \{p_1(x_1,y_1,z_1), p_2(x_2,y_2,z_2), \ldots.., p_n(x_n,y_n,z_n)\} \ldots\ldots\ldots\ldots \text{(eqn 3)}$$

To solve the problem of the 'curse of dimensionality' mentioned previously, first irrelevant dimensions of C are removed, as they may obfuscate robust cluster detection. Then the DBSCAN algorithm is applied. Removing irrelevant dimensions is a well-studied technique (e.g., Parsons et al. 2004). In the case herein, for each cube a pair of different sets of two-dimensional points will be generated. The first one {XZ} is where the dimension Y is removed from the points of the cube C. The second set {YZ} is where the dimension X is removed from the points of C, as shown in equations (eqn 4 and eqn 5).

$$\{XZ\} = \{p_{\cdot 1}(x_1,z_1), p_{\cdot 2}(x_2,z_2), \ldots.., p_{\cdot n}(x_n,z_n)\} \ldots\ldots\ldots \text{(eqn 4)}$$

$$\{YZ\} = \{p_{\cdot\cdot 1}(y_1,z_1), p_{\cdot\cdot 2}(y_2,z_2), \ldots.., p_{\cdot\cdot n}(y_n,z_n)\} \ldots\ldots\ldots \text{(eqn 5)}$$

Once the irrelevant dimensions have been removed, the DBSCAN algorithm is applied to the two sets {XZ} and {YZ} according to the following algorithm:

```
noise = DBSCAN ({XZ}, Eps, MinPts)
if (noise / PTS) <= maxNoise) {
        C is dense cube
}else{
        noise = DBSCAN ({YZ}, Eps, MinPts)
        if (noise / PTS) <= maxNoise){
        C is dense cube
        }else{
        C is sparse cube
        }
}
```

Since DBSCAN is an integer method, it returns the number of noise points in C. PTS is the total number of points in C. The term maxNoise is the maximum allowed percentage of points to consider whether the cube in question is dense or sparse. The value of this percentage is defined in an empirical way, as will be explained in the next section. This is to say, only the smallest number

of noise points of the two sets {XZ} or {YZ} is used, because the minimum noise indicates the better clustering between the two directions. If the noise in one of the views is less than the percentage maxNoise, then one may conclude that C is a dense cube. Otherwise, if both minimum noise values exceed the maxNoise, then C is classified as sparse cube. To reduce the computational time, first the noise is calculated in one direction. If it is less than the maxNoise, then there is no need to calculate the same for the second direction. Otherwise calculation proceeds.

In Figure 3, a considerable distance reduction between the points is visible when the irrelevant dimensions Y and X have been removed from the corresponding views. Although removing dimensions is useful in the case of the X-view and Y-view, the same approach is not applicable in the case of the Z-view (Top). As shown in Figure 3, there are a lot of holes between the points from the TOP-view. For this reason, this view is not considered in the approach presented herein.

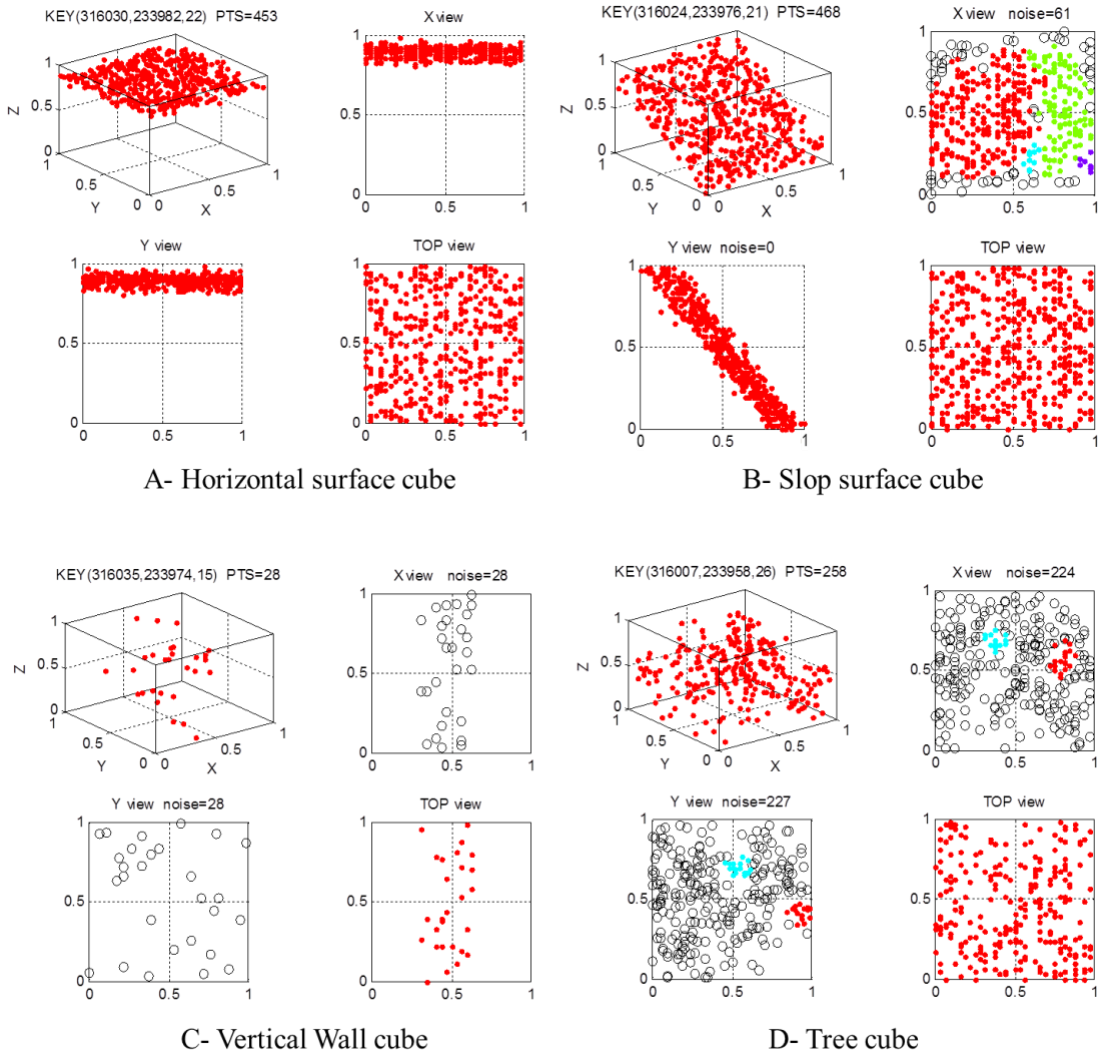


Figure 3. Some of the representative cubes.

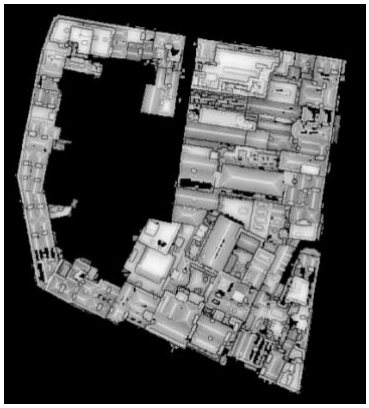
Figure 3 also illustrates some of the representative cubes of the DSM. Figure 3-A shows a prismatic cube with the key identification KEY(316030,233982,22) and 453 points. The cube consists of a horizontal surface, which represents a subspace of a building roof with a height equal to 22 m. Within the cube's prismatic shape, there are some holes between the points. However, when the Y-dimension and the X-dimension are removed in the X-view and Y-view respectively, clusters are readily visible. This is to say, the X-view is the projection to the XZ plane, and Y-view is the projection to the YZ plane. In this case, the noise in the two views is equal to zero, so that this cube is classified as a dense cube. Noise points are presented in the Figure as small circles. The cube in Figure 3-B is a sloped building roof. This cube is also classified as dense, although the noise in the X-view is significant (noise = 113 points), because the noise point number in the Y-view is 0. The cube in Figure 3-C represents a vertical building wall. This cube is classified as sparse, because neither of the two views forms a cluster. The percentage of the noise points in the two views exceeds that of the maxNoise. As previously noted by Jochem et al. (2006) the relative position of the aircraft to the building wall and the large incidence angles often limits point acquisition on these vertical surface, which could affect negatively the results, because the cubes from vertical walls are formed with relatively low point densities.

Notably, a general move to higher point densities and new means of flight path planning to maximize vertical data capture (e.g., Hinks et al. 2009) should help to mitigate this problem. Furthermore, this limitation when it generates a gap could be used to identify a separation between the ground surface and the building roofs, further facilitating roof classification, localization, and then the extraction. Figure 3-D shows a cube that represents vegetation or other undesirable points. No cluster emerges in either of the two views. So this cube is classified as a sparse cube, because the percentage of noise in the both views is greater than the allowed percentage.

Step-3: Clustering interesting cubes

Once the two cube types are generated, the sparse cubes are removed. The remaining dense cubes are grouped into clusters, with each cluster representing the outline of an object. To do that, the Neighbour Adjacent Cube Algorithm is applied. In that algorithm, two adjacent cubes belong to the same object. This algorithm starts from the highest cube in the dataset and moves downwards towards the lowest one. As previously mentioned, because vertical walls cubes are not classified as dense cubes, empty spaces form between the roof building cubes and the ground cubes. Once the highest object is segregated, the algorithm next segregates the highest remaining object in the DSM. This continues, until all objects are segregated from the DSM. Notably, although the algorithm always starts with the highest object, this does not require a height calculation of each object, only determination as to which cube has the highest Z coordinate. This is an improvement over other

346 related works [e.g., Zhang et al. (2006), Abdullah at el. (2014), and Aljumaily at el. (2015)] where
347 object height calculation was needed prior to extraction.
348 At the end of the classification and then the extraction processes, the DSM dataset being
349 processed will be empty, because all of its cubes will have been segregated and moved
350 automatically to the corresponding files where each file represents a cluster. A limitation of this
351 approach is that if two buildings are joined together (see Figure 4) [e.g., terraced housing], the
352 approach recognizes them as a single object. This is a well-known problem for many techniques
353 attempting individual building extraction, as reported by Truong-Hong and Laefer (2015).
354



355
356 Figure 4. An object with multiples adjacent buildings.

357 **EXPERIMENTS AND RESULTS**
358 The clustering approach presented in Section 3 is evaluated herein on an architecturally dense and
359 complex portion of Dublin, Ireland with the aforementioned 225pts/m^2 data. Within the 1 km^2 study
360 area, there are 9 tiles from the DSM (see Figure 5). The DSM preparation included several steps
361 that are outside of the scope of this work, including flight path planning, as well as data collection,
362 registration, and filtering (as described in Truong-Hong, 2011).
363

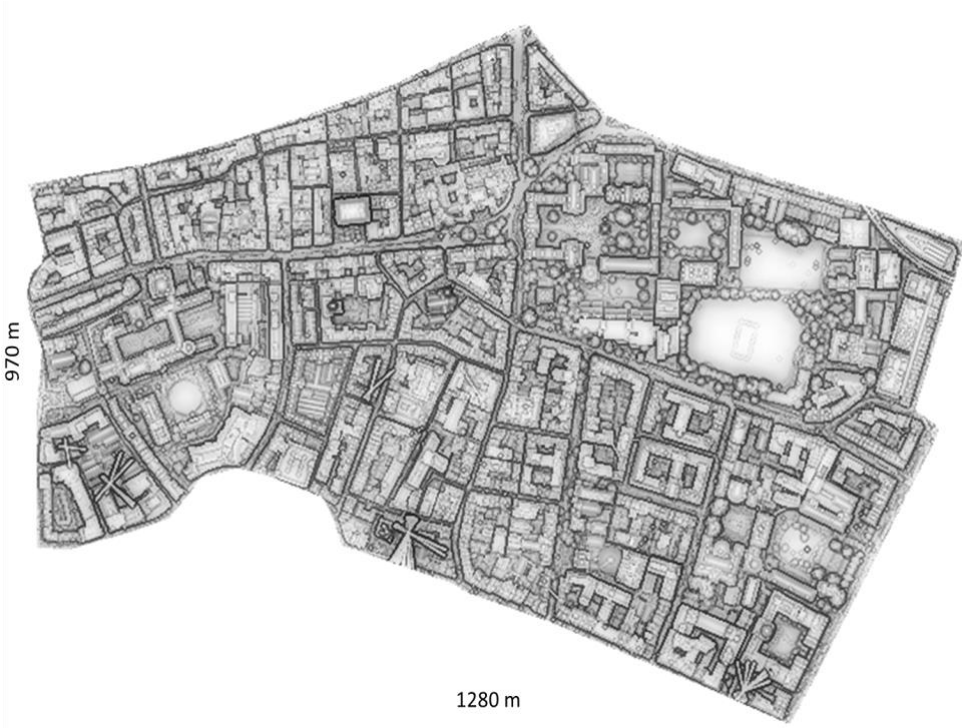


Figure 5. DSM of the study area generated herein

The proposed approach depends on the point cloud density. This DSM contained 225,793,264 points according to the first step of the approach (Step-1). These points were mapped to 3,248,520 cubes resulting in an average of 70 points per cube with a standard deviation of 105.5 points. Their distribution is shown in Figure 6, where the x-axis shows the density divided into 10 point intervals (e.g., the first represents cubes containing 1-10 points). In the last interval, there was only 1 cube, which contained 666 points, which was the densest in the data set. The peak of the histogram represents that 3.95% of the points of the DSM. Those cubes contained 241-250 points.

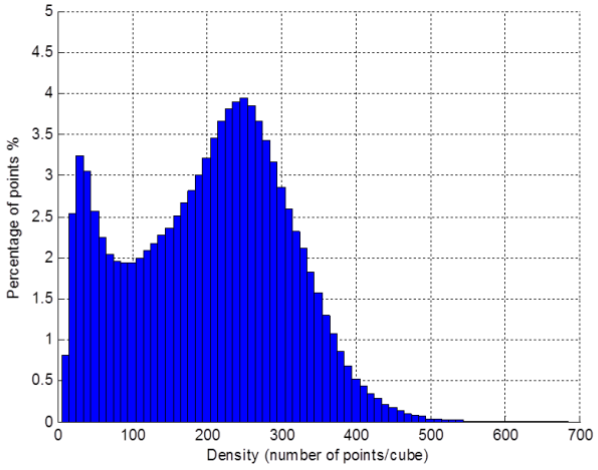


Figure 6. Distribution percentages of the DSM original

To quantitatively evaluate the approach's outputs, the following metrics were used: (1) measurements of correctness, (2) completeness and (3) fitness measure (F-measure). Many

researchers use the terms precision and correctness interchangeably. The same is true for the terms recall and completeness. In this paper the terms correctness and completeness will be used to be consistent with the authors' previously published work (i.e. Aljumaily et al. 2015). Normally, these measures are calculated by taking the difference between the classified objects and the reference objects (Maurya et al. 2012). According to Rutzinger et al. (2009), the correctness evaluates the exactness of the approach and is a ratio of the relevant points of a classified object to the total number of points of the referenced objects (see eqn 6). A point was considered relevant, if the approach classified it correctly, with respect to the corresponding reference object. The completeness is the ratio of the classified relevant points to the total number of points in the referenced objects (see eqn 7). The F-measure or fitness evaluates the overall quality (see eqn 8). These metrics are calculated using terms such as True Positives (TP), False Positives (FP), and False Negatives (FN). TP are the points correctly included into this object, FP are the points incorrectly included into this object, and FN are the points mistakenly excluded for this object.

$$\text{correctness} = \frac{TP}{TP + FP} \quad \text{(eqn 6)}$$

$$\text{completeness} = \frac{TP}{TP + FN} \quad \text{(eqn 7)}$$

$$\text{F-measure} = \frac{TP}{TP + FP + FN} \quad \text{(eqn 8)}$$

As mentioned in the previous section, in order to apply the DBSCAN algorithm several parameters (i.e. Eps and MinPts) must be set. Because these parameters depend on the features of the dataset, the accuracy of the resulting clustering is directly depending on the user's choice of parameters (Zhou et. al. 2012). For this reason, an empirical study was undertaken on the selected DSM to optimize these parameters for optimal clustering results. In this study, initial values for each parameters were assigned. These were then increased incrementally and individually until a degradation of the results was observed. A global optimisation was not, however, undertaken. The initial values of Eps and MinPts were selected as 0.1m and 10 points, respectively based on the belief that 0.1 m would be a reasonable distance between two points in a DSM to form a cluster, and a cluster with less than 10 points having insufficient information for meaningful post-processing activities. The values of Eps and MinPts were increased by 0.05m and 10 points, respectively. In addition, maxNoise was used as a third parameter to distinguish between a classification of dense and sparse. In the same way an initial value of (maxNoise < 10) was selected as the division between dense and sparse cubes. This is to say, if the percentage of the noise in a cube was less than or equal to 10%, then this cube was considered as a dense cube but otherwise as sparse cube. An initial setting of 0% only resulted in a clustering of 7% of the data, thus the selected

lowerbound threshold was 10%. The incremental value of maxNoise was established as 10% for each iteration (i.e., 10%, 20%, 30%, etc). The most significant results of this study are shown in Table 1, where the bolded values represent the most interesting results of the approach. As shown in Table 1, many results are similar often achieving around 85% correctness despite being derived from different thresholds; with the remaining 15% likely to have been lost because of the complexity of the architecture of the building and the difficulties related to the manual extraction of the reference objects. Clearly, a further development step is needed in which a formal optimization process is undertaken. With this caveat understood, a single case will be further investigated below. Due to space limitations only one set of results is further presented (Eps = 0.1m, MinPts=10, and maxNoise<30%). Those achieved nearly 85% in all three categories: correctness = 84.91, completeness = 84.39, quality = 84.65 (Table 1).

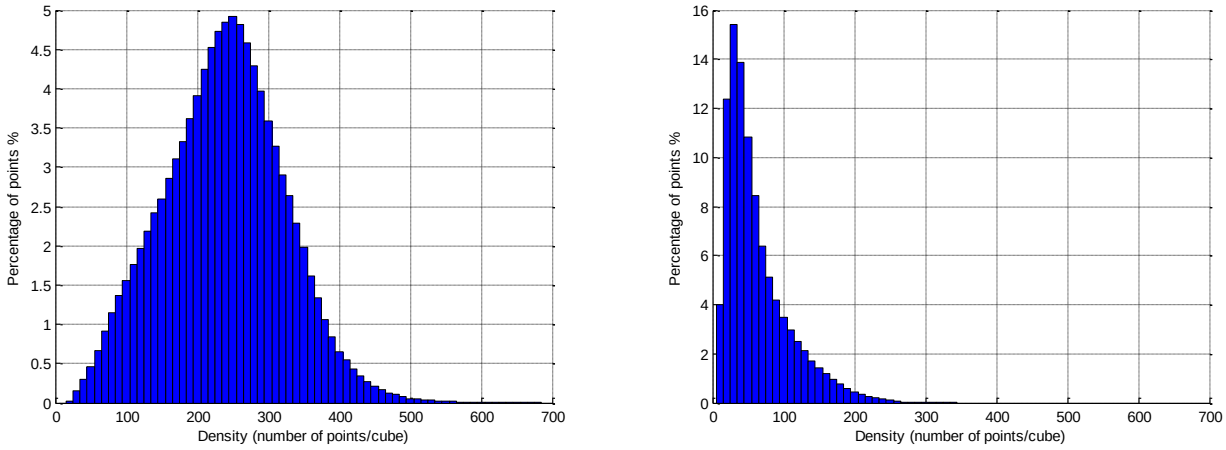
Figure 7 illustrates the analysis of the clustering results, where two sets of points can be distinguished. The first are the points of the dense cubes (Figure 7-A) representing approximately 80% of the DSM across 30% of the cubes, with an average density of approximately 185 pts/cube and a standard deviation of 85.0. They display a normal distribution. The second are the points of the sparse cubes (Figure 7-b) representing approximately 20% of the DSM across 70% of the cubes with an average density of approximately 20 pts/cube and a standard deviation of 44.6 displaying a normal distribution. In these figures, there is a separation between cubes with high point density and those with low point density. Those with a density of 250 points/cube are definitely dense categorization and those with a density of less than 100 points per cube definitively sparse. Looking at Figure 7, most of the cubes that contained between 100 and 250 points per cube were classified ultimately by the proposed approach as dense cubes.

Table 1. The most significant results of the empirical study.

maxNoise	***MinPts***	***Eps (m)***	***Correctness***	***Completeness***	***Quality***
10%	10	0.1	78.06	90.05	83.62
		0.15	**84.51**	**85.40**	**84.96**
		0.2	87.19	76.88	81.71
	20	0.1	65.73	96.41	78.17
		0.15	77.22	90.99	83.54
		0.2	**81.44**	**88.13**	**84.65**
20%	10	0.1	**81.68**	**87.57**	**84.52**
		0.15	87.66	79.86	83.58
		0.2	87.14	51.37	64.63

	20	0.1	70.33	94.26	80.56
		0.15	**80.68**	**88.69**	**84.50**
		0.2	**84.39**	**83.99**	**84.19**
30%	10	0.1	**84.91**	**84.39**	**84.65**
		0.15	87.87	71.50	78.85
		0.2	87.20	30.40	45.09
	20	0.1	75.48	92.21	83.01
		0.15	**83.41**	**86.13**	**84.75**
		0.2	86.84	79.32	82.91
40%	10	0.1	**85.73**	**81.73**	**83.68**
		0.15	87.84	60.45	71.61
		0.2	90.14	18.86	31.20
	20	0.1	77.76	90.86	83.80
		0.15	**84.17**	**83.56**	**83.87**
		0.2	87.71	78.00	82.57

435

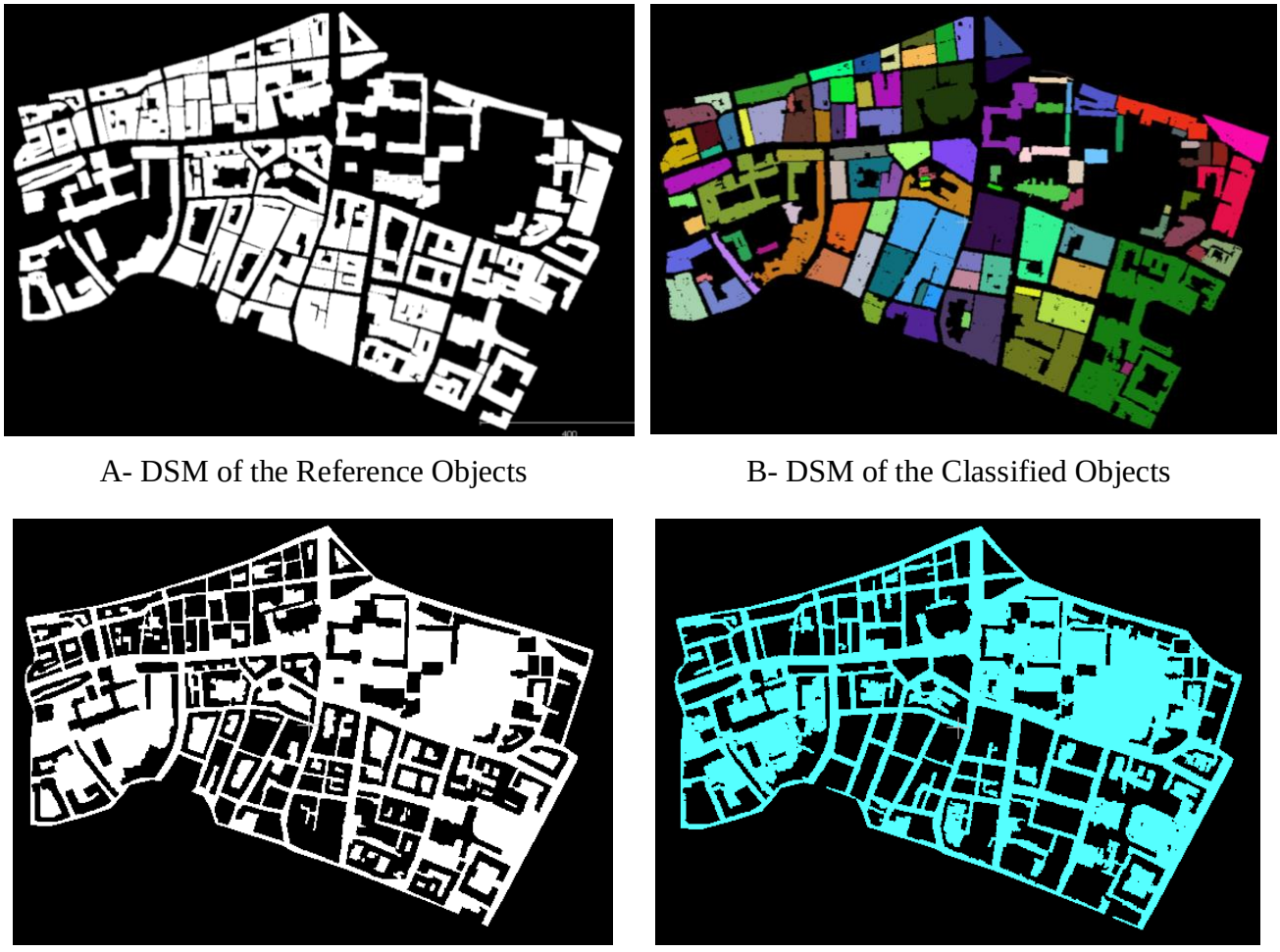


A- Dense Cubes B- Sparse Cubes

436 Figure 7. Clustering classification results for Eps = 0.1m, MinPts=10, and maxNoise<30%

437 Figure 8 shows a comparison study between the reference objects and those automatically
438 classified objects by the approach proposed herein. First, the reference objects (Figure 8-A) were
439 manually extracted from the original DSM. In total, there were 106 building groups and 1 ground
440 surface. This was done using the visualization tool CloudCompare (Compare, 2015); most features
441 within the study area (e.g., buildings, roads, trees) were easily distinguishable with the naked eye.
442 CloudCompare provides editing features such as DSM segmentation. Each colour in Figure (8-B)
443 represents a classified object. Giving a colour to an object means that the object was defined, and all

444 the relevant information about it (e.g., coordinates, dimensions, outlines) was harvested to enable
445 simple extraction of the whole object from the original DSM.
446 For simplicity, the previous parameters (TP, FP, and FN) were calculated only considering
447 the cube level. This is to say, all the cubes that were included in both DSMs (Figure 8-A and 8-B)
448 were considered as TP, if all the cubes that were included in the DSM are classified correctly as
449 objects (Figure 8-B). They were FP if they classified an object that did not appear as a reference
450 objects (Figure 8-A). Finally, all cubes that were included in the DSM as reference objects (Figure
451 8-A) but not classified objects (Figure 8-B) were considered as FN. The same approach was taken
452 with respect to the classification of the ground surface (Figure 8-C and 8-D).



A- DSM of the Reference Objects

B- DSM of the Classified Objects

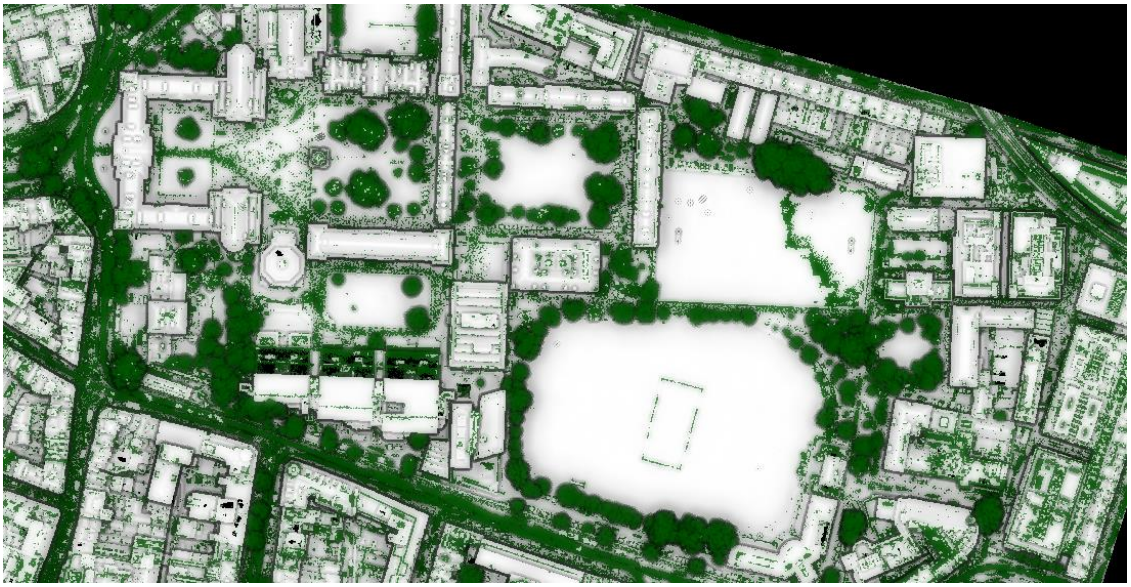
C- The reference of the ground surface

D- The cluster of the classified ground surface

453 Figure 8. Quantitative evaluate of the approach's outputs for Eps = 0.1m, MinPts=10, and
454 maxNoise<30%.

455 As visible in Figure 9, the approach was able to classify the heavy vegetation and other
456 obstructions in the DSM and able to segregate fully automatically the outlines of building groups
457 from the roads and the ground surface. The approach was also very successful for classifying

458 building groups with complicated roof geometries and those with multiple sections of varying
459 heights.
460



461

462 Figure 9. Vegetation detection by the proposed approach.

463 In order to validate the results of the proposed approach, a new DSM of approximately 1
464 km^2 was considered (Figure 10). This new data set was selected also from an architecturally dense
465 and complex portion of Dublin, Ireland. The new data set contained 158,890,014 points. These
466 points were mapped to 2,150,979 cubes. This DSM was selected, because it is more complicated
467 than the first one, where many small objects with vegetation and other obstacles were included.
468

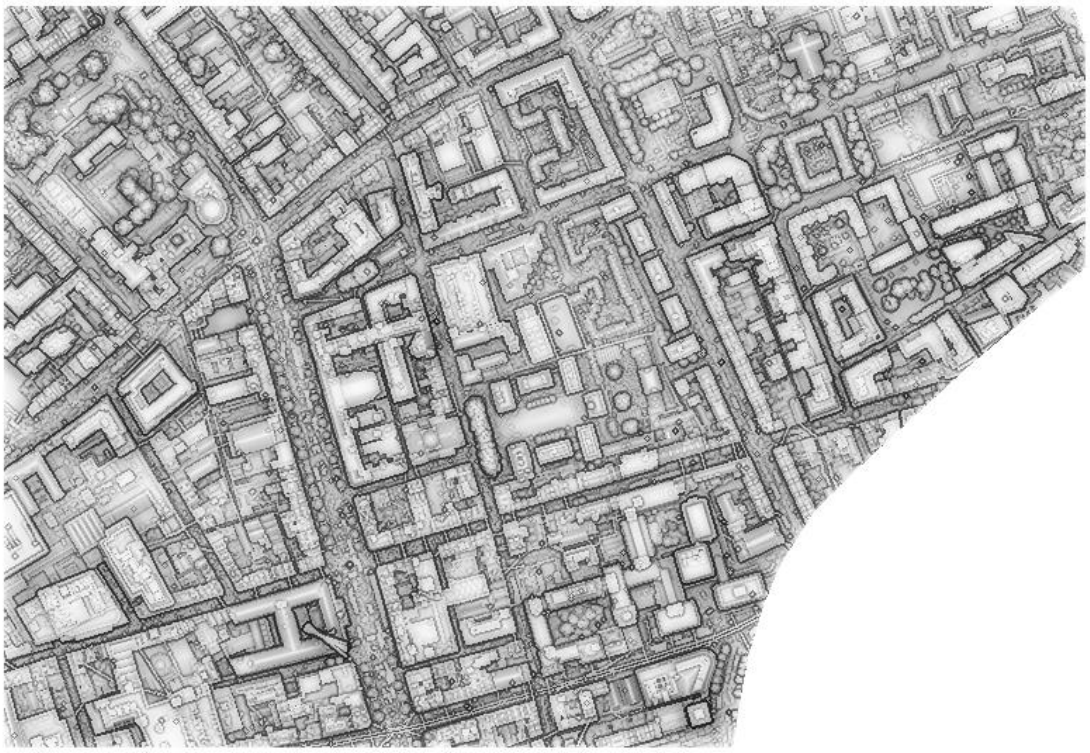
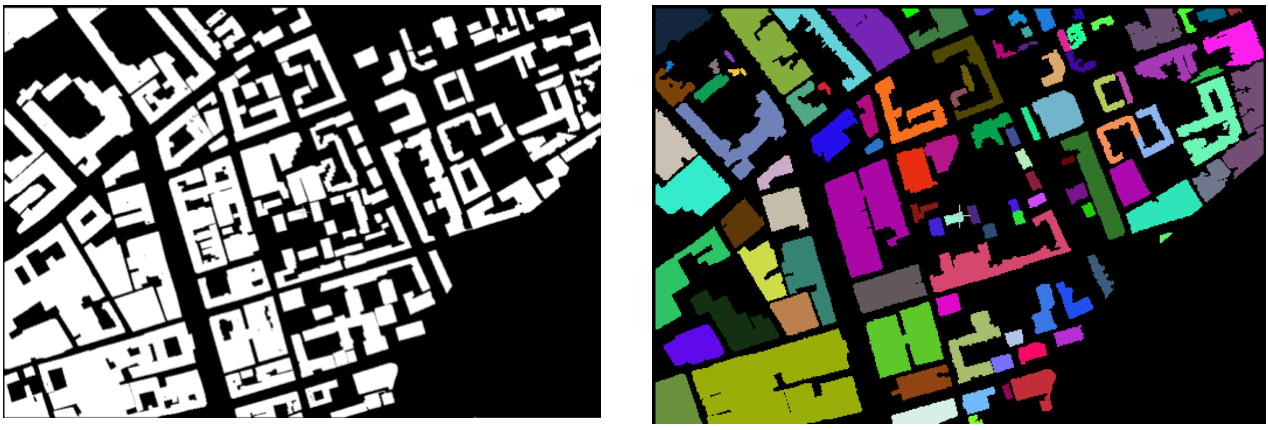


Figure 10. DSM of the second study area generated herein

In the validation of the second study area the same thresholds were selected (Eps=0.1m, MinPts=10, and maxNoise < 30%. The results of the extraction quality were slightly poorer (correctness=80.02%, completeness=78.01%, and overall quality = 79.44%). The difference between the qualities of the both study areas is in part an outgrowth of the difficulty of manual extraction of the second study area where errors may be introduced. In Figure 11, the DSM of the classified objects and the referenced objects are shown.



A- DSM of the Reference Objects B- DSM of the Classified Objects

Figure 11. Quantitative evaluate of the approach's outputs of the second case study.

The computational efficiency and scalability of the approach needs to be demonstrated with respect to the execution time required for each step of the proposed approach. All the experiments were conducted on a PC with an Intel Core i7 CPU 2.40GHz, 12GB Memory. Regarding to Step-1

(MapReduce Step) the experiment was done using a single node of a Hadoop installation. The results of execution time are shown in Figure 12. In this figure, the experiments were done by selecting three different cube dimensions (0.5m, 1.0m, and 2.0m) of the same DSM (Figure 5). Notably, the execution time using cubes with dimensions of 0.5m were very similar to those using dimensions of 1.0m. A value of d of 1.0m is recommended, because when the cube dimension is small and there are many cubes of low density with complex architecture, the original cluster will unnecessarily be divided into smaller clusters and this will reduce the quality of the clustering. If a big cube dimension such as 2.0m is selected, other problems arise. The main problem is that when the cube dimension is large, a cube may contain parts of different objects including vegetation, which results in their ultimate misclassification. In addition, big cube dimensions mean that there are a high number of points to be processed within each cube. Consequently, extensive execution time will be need in the next steps.

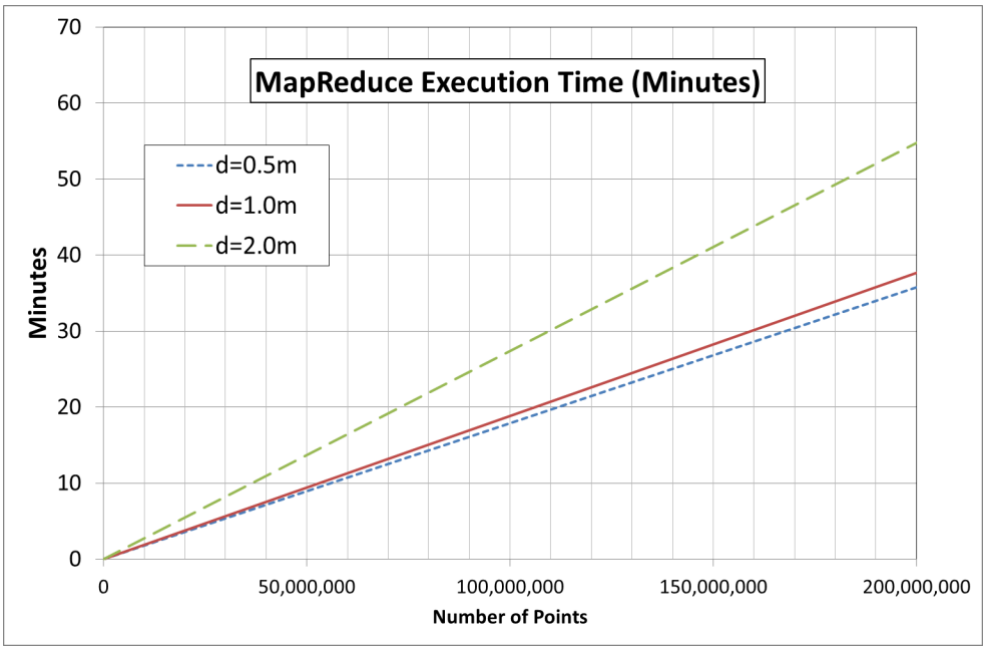


Figure 12. Execution time of the MapReduce step with three different dimensions cubes

Notably, the execution time of the MapReduce step can be improved easily by adding more nodes as is common for a Big Data platform cluster (Xu et al. 2015). According to the result showed in Wang et al. (2010) using a cluster of 8 nodes a MapReduce based spatial system consumes approximately 0.24% of the initial time.

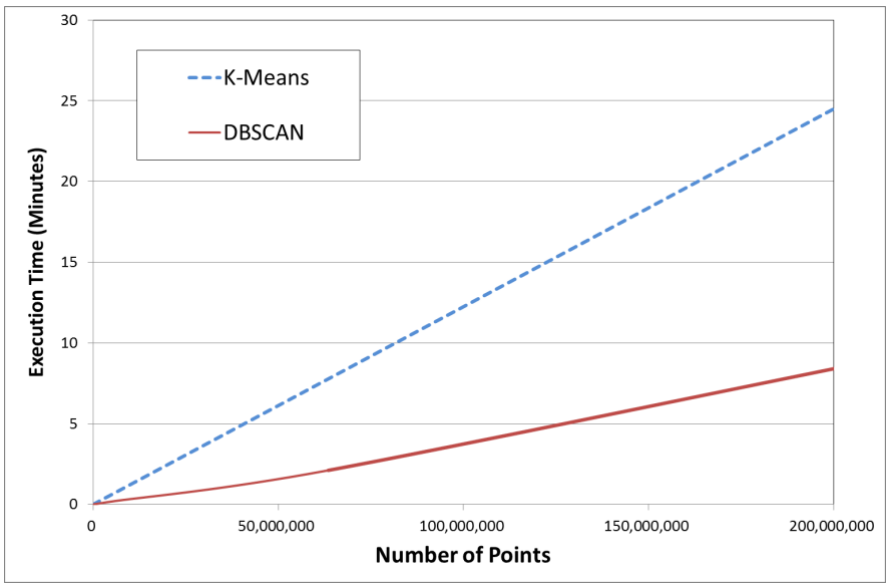


Figure 13. Comparison between the proposed execution time for DBSCAN versus K-Means time

Regarding to the second step (Recognizing interesting subspaces), as mentioned in the related works section, the DBSCAN is considered a classic clustering algorithm based on density. The complexity of this algorithm is O(n*logn), and it is robust towards outlier detection (Noise) (Xu et al. 2015). Other classic algorithm such as K-means is not built for outliers' detection purpose. However, in this section in order to ensure the validity of the performance of the DBSCAN algorithm, a comparison between DBSCAN and the K-Means has been done. But firstly, the K-Means algorithm has been adapted for our approach, i.e., improving K-Means in order to detect outliers according to the approach presented into (McCaffrey 2013). The result of the comparison is shown in Figure 13, where it is clearly shown that the DBSCAN algorithm provides better performance than the K-Means algorithm. For example, in order to cluster 200,000,000 points the DBSCAN needs no more 9 minutes while the K-Means needs approximately 25 minutes. The difference likely stems from the fact that DBSCAN was built to detect outliers, while the K-Means was built for space partitioning, with outlier detection and removal as added components (e.g., Hautamäki, et. al. 2005).

The time for Step-3 (Clustering interesting cubes) of the approach is insignificant when compared with the time needed for the first two phases. The execution time of Step-3 (see Figure 14) depends on the size of the object extracted. The object classification time depends on the number of affiliated and adjacent points. For example, in the case presented herein, each of the classified clusters (73%) had less than 1,000,000 points. So, to cluster interesting cubes of an object with 1 million points, this Step-3 needed approximately 2,000 milliseconds (see Figure 14).

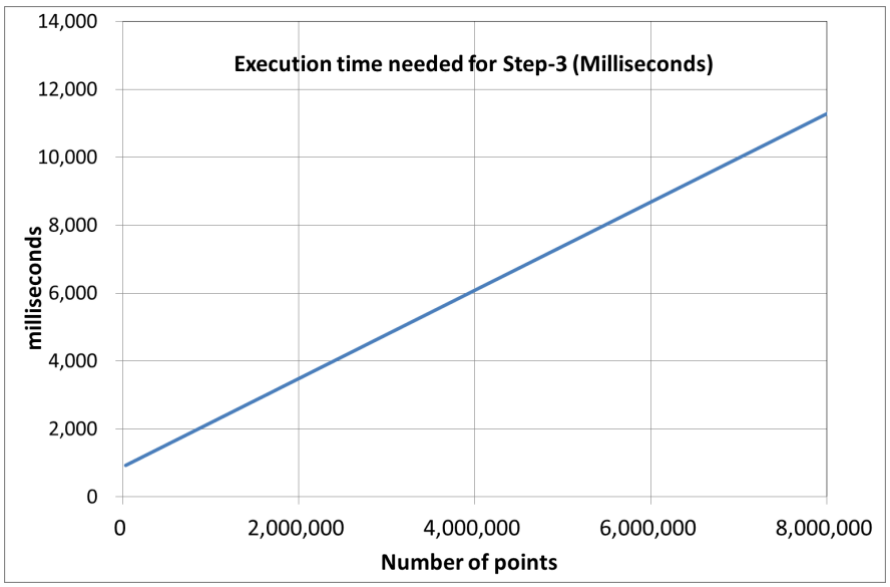


Figure 14. Execution time of the clustering interesting cubes step of the approach

CONCLUSIONS

Most current solutions for building classification and extraction from point cloud data are hard to scale with respect to the forward trajectory of data densities levels. To address this in the context of object classification, localization and extraction, this paper presents the implementation of a fully automatic and significantly more scalable approach than otherwise available to support clustering of LiDAR data in a Big Data context. The approach was tested with two study areas on approximately 2 km^2 where more than 200 objects in the DSM were automatically detected at an average classification quality level of 85% including those with complicated roof geometries and those with multiple sections of varying heights. The approach was able to classify the heavy vegetation and other obstructions in the DSM and was able to segregate fully and automatically the outlines of buildings from the roads and ground surface.

According to the obtained results, this paper presents the viability to use a clustering algorithm (DBSCAN) based on point density for the objects extraction from a DSM. This kind of clustering is suitable for data with arbitrary shape and is able to discard noise or outliers (in this case, points that do not belong to a building or road). While the current implementation DBSCAN algorithm has the drawbacks of requiring a high memory capacity with big volume data, the Big Data framework where the approach is being developed will largely alleviates this problem. In the ongoing work, the issue of a global optimization strategy for automated parameter selection will also be considered.

ACKNOWLEDGMENTS

This work was in part supported with funds from the European Research Council Project 307836.

549

REFERENCES

Abdullah, S., Awrangjeb, M., and Lu, G. Automatic segmentation of LiDAR point cloud data at difference height levels for 3D building extraction. *IEEE International conference on Multimedia and Expo workshops* (ICMEW) (Chengdu, 2014) (pages 1-6).

Aggarwal, C. C., & Reddy, C. K. (Eds.). (2013). *Data clustering: algorithms and applications*. CRC Press.

Aljumaily, H., Laefer, D. F., & Cuadra, D. (2015). Big-Data Approach for Three-Dimensional Building Extraction from Aerial Laser Scanning. *Journal of Computing in Civil Engineering*, *30*(3), 04015049.

Awrangjeb, M., Zhang, C., & Fraser, C. S. (2013). Automatic extraction of building roofs using LiDAR data and multispectral imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, Volume 83, September 2013, Pages 1-18.

Compare, C. (2015). 3D point cloud and mesh processing software Open Source Project. License: GNU GPL (General Public Licence), Version: 2.6, http://www.danielgm.net/cc

Chang, J. W. (2005, September). A new cell-based clustering method for high-dimensional data mining applications. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems* (pp. 391-397). Springer Berlin Heidelberg.

Chang, J. W., & Jin, D. S. (2002, March). A new cell-based clustering method for large, high-dimensional data in data mining applications. In *Proceedings of the 2002 ACM symposium on Applied computing* (pp. 503-507). ACM.

Chakrawarty, L., & Gupta, P. (2014) Applying SR-Tree technique in DBSCAN clustering algorithm. International Journal of Application or Innovation in Engineering & Management. Volume 3, Issue 1, January 2014, pages 207-210.

Chen, Y., Crawford, M., & Ghosh, J. (2006, July). Improved nonlinear manifold learning for land cover classification via intelligent landmark selection. In *2006 IEEE International Symposium on Geoscience and Remote Sensing* (pp. 545-548). IEEE.

Darong, H., & Peng, W. (2012). Grid-based DBSCAN algorithm with referential parameters. *Physics Procedia*, *24*, 1166-1170.

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).

Fu, X., Hu, S., & Wang, Y. (2014). Research of parallel DBSCAN clustering algorithm based on MapReduce. *International Journal of Database Theory and Application*, *7*(3), 41-48.

Hautamäki, V., Cherednichenko, S., Kärkkäinen, I., Kinnunen, T., & Fränti, P. (2005, June). Improving k-means by outlier removal. In *Scandinavian Conference on Image Analysis* (pp. 978-987). Springer Berlin Heidelberg.

585 Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern recognition letters*, *31*(8),
586 651-666.
587 Jochem, A., Höfle, B., & Rutzinger, M. (2011). Extraction of vertical walls from mobile laser
588 scanning data for solar potential assessment. *Remote sensing*, *3*(4), 650-667.
589 Jochem, A., Höfle, B., Wichmann, V., Rutzinger, M., Zipf, (2012) A. Area-wide roof plane
590 segmentation in airborne LiDAR point clouds. Comput. Environ. Urban Syst. 2012, 36, 54-64.
591 Kailing, K., Kriegel, H. P., & Kröger, P. (2004, April). Density-connected subspace clustering for
592 high-dimensional data. In *Proc. SDM* (Vol. 4). Pages 246-256
593 Kurasova, O., Marcinkevicius, V., Medvedev, V., Rapecka, A., & Stefanovic, P. (2014, November).
594 Strategies for big data clustering. In *2014 IEEE 26th International Conference on Tools with*
595 *Artificial Intelligence* (pp. 740-747). IEEE.
596 Lee, I., Cai, G., & Lee, K. (2014). Exploration of geo-tagged photos through data mining
597 approaches. *Expert Systems with Applications*, *41*(2), 397-405.
598 Li, Y., Wu, H., An, R., Xu, H., He, Q., & Xu, J. (2013). An improved building boundary extraction
599 algorithm based on fusion of optical imagery and LiDAR data. *Optik-International Journal for*
600 *Light and Electron Optics*, *124*(22), 5357-5362.
601 Maurya, R., Gupta, P. R., Shukla, A. S., & Sharma, M. K. (2012, March). Building extraction from
602 very high resolution multispectral images using NDVI based segmentation and morphological
603 operators. In *Advances in Engineering, Science and Management (ICAESM), 2012 International*
604 *Conference on* (pp. 577-581). IEEE.
605 McCaffrey J. (February 2013) Data Clustering - Detecting Abnormal Data Using k-Means
606 Clustering. In *The Microsoft journal for developers*. https://msdn.microsoft.com/en-
607 us/magazine/jj891054.aspx
608 Parsons, L., Haque, E., & Liu, H. (2004). Subspace clustering for high dimensional data: a review.
609 *ACM SIGKDD Explorations Newsletter*, *6*(1), 90-105.
610 Rottensteiner, F., Trinder, J., Clode, S., & Kubik, K. (2004). Fusing airborne laser scanner data and
611 aerial imagery for the automatic extraction of buildings in densely built-up areas. *International*
612 *Archives of Photogrammetry and Remote Sensing*, *35*(B3), 512-517.
613 Rutzinger, M., Rottensteiner, F., & Pfeifer, N. (2009). A comparison of evaluation techniques for
614 building extraction from airborne laser scanning. *IEEE Journal of Selected Topics in Applied*
615 *Earth Observations and Remote Sensing*, *2*(1), 11-20.
616 Slatton, K.C., and Carter,W.E.,2008,A Primer for Airborne LiDAR. The National Center for
617 Airborne Laser Mapping (NCALM)
618 http://ncalm.cive.uh.edu/sites/ncalm.cive.uh.edu/files/files/publications/reports/LidarPrimer-
619 Final02.pdf

Truong-Hong, L. (2011). Automatic Generation of Solid Models of Building Facades from LiDAR Data for Computational Modelling. , Ph. D. Thesis, University College Dublin

Truong-Hong, L., Laefer, D. F., Hinks, T., & Carr, H. (2013). Combining an angle criterion with voxelization and the flying voxel method in reconstructing building models from LiDAR data. Computer-Aided Civil and Infrastructure Engineering, 28(2), 112-129.

Truong-Hong, L., Laefer**,** D. (2015). "Quantitative Evaluation Strategies for Urban 3D Model Generation from Remote Sensing Data." Computers and Graphics, Elsevier, doi.org/10.1016/j.cag.2015.03.001i.

UCD Digital Library, (2007) http://dx.doi.org/10.7925/DRS1.UCDLIB_30462

Vo, A. V., Truong-Hong, L., Laefer, D. F., & Bertolotto, M. (2015). Octree-based region growing for point cloud segmentation. ISPRS Journal of Photogrammetry and Remote Sensing, 104, 88-100.

Wang et al. 2010, Dec. Accelerating spatial data processing with mapreduce. Parallel & Distrib. Sys. IEEE pages 229-236.

Wang, C., Hu, X., Ji, M., & Li, T. (2013). An Automated 3D Approach for Buildings Reconstruction from Airborne Laser Scanning Data. In *Geo-Informatics in Resource Management and Sustainable Ecosystem* (pp. 704-712). Springer Berlin Heidelberg.

Wu, Y. P., Guo, J. J., & Zhang, X. J. (2007, August). A linear dbscan algorithm based on lsh. In *Machine Learning and Cybernetics, 2007 International Conference on* (Vol. 5, pp. 2608-2614). IEEE.

Xu, D., & Tian, Y. (2015). A comprehensive survey of clustering algorithms. Annals of Data Science, 2(2), 165-193.

Xu, G., Yu, W., Chen, Z., Zhang, H., Moulema, P., Fu, X., & Lu, C. (2015). A cloud computing based system for cyber security management. *International Journal of Parallel, Emergent and Distributed Systems*, *30*(1), 29-45.

Zhang et al. 2009, Aug. Spatial queries evaluation with mapreduce. Grid & Coop Comp 2009, 287-92. IEEE

Zhang, K., Yan, J., & Chen, S. C. (2006). Automatic construction of building footprints from airborne LíDAR data. *IEEE Transactions on Geoscience and Remote Sensing*, , *44*(9), pages 2523-2533.

Zhou, H., Wang, P., & Li, H. (2012). Research on adaptive parameters determination in DBSCAN algorithm. *Journal of Information & Computational Science*, *9*(7), 1967-1973.

Zhou, X., Xu, C., & Kimmons, B. (2015). Detecting tourism destinations using scalable geospatial analysis based on cloud computing platform. *Computers, Environment and Urban Systems*, *54*, 144-153.