CLASSIFICATION OF BIG POINT CLOUD DATA USING CLOUD COMPUTING

Kun Liu* Jan Boehm

Dept of Civil, Environ & Geomatic Eng, University College London, United Kingdom
{kun.liu, j.boehm}@ucl.ac.uk

Commission III, WG III/5

KEY WORDS: Point cloud, Machine learning, Cloud computing, Big data

ABSTRACT:

Point cloud data plays an significant role in various geospatial applications as it conveys plentiful information which can be used for different types of analysis. Semantic analysis, which is an important one of them, aims to label points as different categories. In machine learning, the problem is called classification. In addition, processing point data is becoming more and more challenging due to the growing data volume. In this paper, we address point data classification in a big data context. The popular cluster computing framework Apache Spark is used through the experiments and the promising results suggests a great potential of Apache Spark for large-scale point data processing.

1 INTRODUCTION

Point cloud data is increasingly convenient to obtain due to the rapid development of remote sensing technologies such as UAV-based photogrammetry (Pix4D, 2014), indoor mobile mapping (Viametris, 2014), and low-cost consumer RGB-D sensor (Microsoft, 2014). These arising methods and systems provide a variety of feasible means for acquiring scenes with varying scales, and a large volume of data can be generated daily. For example, the AHN2 LiDAR data set (AHN, 2014) covering the whole Netherlands is about half terabyte. However, classical data processing approaches are generally performed on a single machine, they turn out to be not suitable because of the limited computing and storage capacity. Therefore, it is crucial to figure out a solution which is able to process such massive data efficiently.

MapReduce (Dean and Ghemawat, 2008) has been extensively applied in many large-scale applications such as search engine and recommendation system. As one of its alternatives, Apache Spark (Zaharia et al., 2010) has been the most popular cluster computing framework attracting lots of attention from both the industry and the academia. Besides the nature of scalability, Spark also supports fault tolerance and in-memory computing. Noticeably, the latter property endows Spark with remarkable performance, e.g., outperforming Hadoop (White, 2009)—an open source implementation of MapReduce—by 10 times in certain applications (Zaharia et al., 2010). In spite of that, fairly little research has been conducted on massive point cloud data processing using cloud computing technologies, especially Apache Spark.

Filling this gap we present our work on one of the fundamental point cloud processing tasks in the geospatial domain - point cloud classification. While a considerable body of work exists on point cloud classification, for very large datasets the scale of data itself becomes a challenge. In this paper, we address the classification problem in a big data context by applying cloud computing on very large point clouds. Promising experimental results are also provided to demonstrate the possibility to apply cloud computing in large-scale geospatial applications.

The rest of this paper is organized as follows. The method for for tree crown classification is overviewed in Section 2. The algorithm is presented in detail in Section 3 as well as the implementation by means of Apache Spark. In Section 4, the experimental results are analysed and discussed. This work is concluded in Section 5.

2 OVERVIEW

Point classification is a typical machine learning problem, concretely, associating each point with a label. The learning procedure in this paper is inspired by (Weinmann et al., 2014) which aims to interpret point clouds semantically, while our work specializes in indicating whether a point belongs to a tree crown. Similar to (Weinmann et al., 2014), our work applies machine learning to achieve the classified results. The classifier is trained using random forest (Breiman, 2001) which is a multitude of decision trees. Point features for machine learning are computed based on the method proposed in (Demantké et al., 2011). Seven different features (Weinmann et al., 2014) are used in our classification problem, respectively encoding linearity, planarity, scattering, omnivariance, anisotropy, eigenentropy, and change of curvature.

An important difference between our work and the ones aforementioned is that massive parallelism in our implementation is realized using cloud computing. This endows our method with a significant scalability. The entire experiment is conducted on a cluster launched in Amazon EC2 service (Amazon, 2015). The public available benchmark (Vallet et al., 2015) for urban point cloud analysis is also used to demonstrate the efficiency and the robustness of our method.

3 TREE CROWN CLASSIFICATION

In this section, a tree crown classification method is proposed. First the algorithm using machine learning technique is presented. The implementation based on Apache Spark is discussed in detail afterwards.

3.1 Algorithm

The classification method proposed in this work is formulated as a supervised learning problem (Bishop, 2006). The arising problem is addressed by three steps, namely, feature computation, model training, and prediction.

*Corresponding author.

This contribution has been peer-reviewed.
Editors: M. Brédif, G. Patanè, and T. Dokken
doi:10.5194/isprsarchives-XL-3-W3-553-2015

3.1.1 Feature computation

Features play a fairly important role in machine learning problems. Features with high quality can simplify learning models to interpret the models more easily and enhance algorithm performance with respect to both the speed and the accuracy.

In our problem, we aim to efficiently differentiate tree crown points and other points by means of features. Points in the neighborhood of a tree crown point tend to be scattered uniformly along all rays from the central point, which indicates a homogeneous distribution. On the other hand, other points generally reveal either the nature of 2D planes such as building facets or the nature of 1D lines such as light poles. In such scenario, eigen values of covariance matrix of points are suitable measures to characterize the dimensional information, which is comprehensively discussed in (Demantké et al., 2011). Concretely, seven 3D features based on eigen values are used through our classification method as shown in Table 1.

Linearity	$(\lambda_1 - \lambda_2)/\lambda_1$
Planarity	$(\lambda_2 - \lambda_3)/\lambda_1$
Scattering	λ_3/λ_1
Omnivariance	$\sqrt[3]{\lambda_1\lambda_2\lambda_3}$
Anisotropy	$(\lambda_1 - \lambda_3)/\lambda_1$
Eigenentropy	$-\sum_{i=1}^{3} \lambda_i \ln \lambda_i$
Change of curvature	$\lambda_3/(\lambda_1 + \lambda_2 + \lambda_3)$

Table 1: 3D features of points

3.1.2 Random forest

Given an input point cloud with computed features described in Table 1 and correct labels, a classifier is trained using random forest. Once the classifier is generated, the prediction process can be conducted on input data with feature information.

Random forest (Breiman, 2001) is a widely applied learning method which can be used for both classification and regression problems. A random forest is a combination of several decision trees (Bishop, 2006) which perform the prediction by traversing the tree structure. Overfitting occurs often within decision trees due to hard value split of each tree node (Bishop, 2006) , which is a significant disadvantage. As an ensemble of decision trees, random forest overcomes such problem by virtue of weighted votes from multiple decision trees. Moreover, random forest also explicitly performs feature selection as each decision tree is created using different random features. Therefore, good features can be selected from a variety of input features so that the prediction accuracy can be improved significantly.

3.2 Implementation

We implement the classification method presented in Section 3.1 by means of Apache Spark (Zaharia et al., 2010) which currently is the most popular cluster computing engine for large-scale data processing. The final results are visualized using Potree(Potree, 2015) which is a WebGL point cloud viewer for large data sets. The entire pipeline is web-based.

3.2.1 Parallel computing in cloud

We utilize Apache Spark to fulfill the parallelization of our method in cloud. Apache Spark is a fast and general-purpose cluster computing library. Similar to Hadoop (White, 2009), it supports the well-known MapReduce (Dean and Ghemawat, 2008) paradigm. In addition, it introduces the resilient distributed dataset (RDD) which can be persisted in memory. This feature can dramatically enhance the performance of Apache Spark over Hadoop especially for applications with iterative operations. It has become the most popular cluster computing system for large-scale data processing in the industry. In Apache Spark, several convenient built-in modules are available including Spark SQL for SQL and structured data processing and MLib for machine learning. The implementation of random forest exists in the module MLib as well. Unlike Hadoop, Apache Spark offers APIs in Java, Scala, Python and R. For point cloud data processing, the Python API is more convenient, due to plenty of existing Python packages for numerical computing and less effort to create a Python binding for C++ libraries. This strength allows us to easily reuse existing libraries in Apache Spark applications. Therefore, the Python API of Apache Spark is used and the entire code for our method is written in Python as well.

As the primary abstraction in Apache Spark, the resilient distributed dataset (RDD) plays a key role to organize data and achieve parallel computation An RDD is simply a list of elements which have same types. Usually each RDD has multiple partitions distributed to cluster nodes and each partition has several copies in different nodes in order to realize the feature of fault tolerance in Apache Spark. Our implementation can be summarized as a series of manipulations of RDDs including creating new RDDs, transforming existing RDDs, and performing operations on RDDs to generate results. In the directed acyclic graph (DAG) as illustrated in Figure 1, the first RDD `RDD_1` is initialized from a list of file name strings through the operation *parallelize*. `RDD_1` is partitioned and is distributed over cluster nodes. Point cloud data is loaded as `RDD_2` by applying *flatmap* on `RDD_1`. `RDD_2` can be regarded as a list of point elements and each element is a 3D vector representing the point coordinate. Features for learning are computed from `RDD_2` and saved as `RDD_3`, and then predicted results are generated by applying pre-trained model on `RDD_3`. The classification results are outputted into file system by performing *groupByKey* and *foreach* on `RDD_5` which is a combination of `RDD_2` and `RDD_4`,

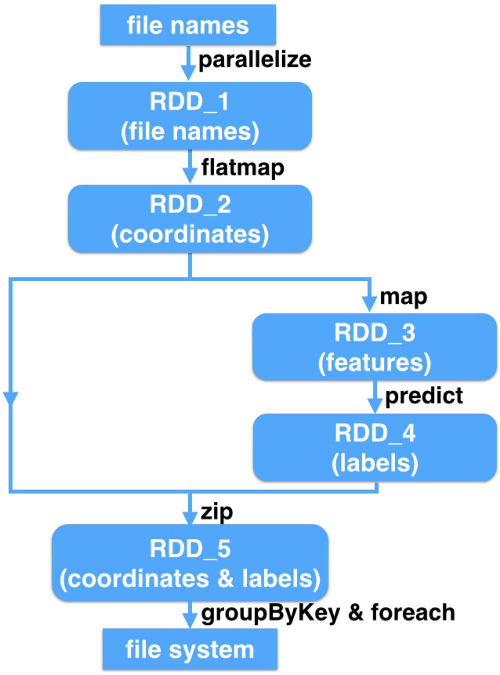
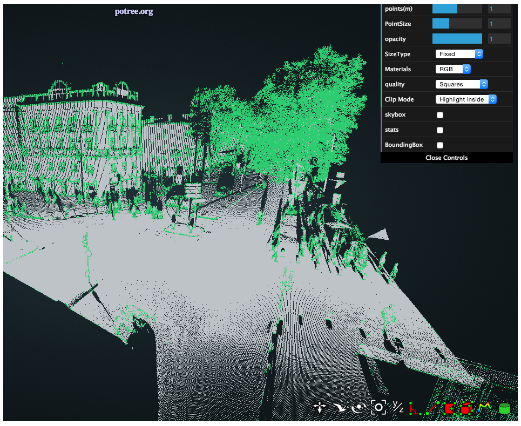


Figure 1: The directed acyclic graph (DAG) in our implementation using Apache Spark.

This contribution has been peer-reviewed.
Editors: M. Brédif, G. Patanè, and T. Dokken
doi:10.5194/isprsarchives-XL-3-W3-553-2015

3.2.2 Visualization

Potree (Potree, 2015) is an open source WebGL based point cloud render especially for large point data sets. As illustrated in Figure 2, its user interface is similar to common computer graphics software, e.g., Blender or Autodesk 3D Max. Since multi-resolution octrees are applied in potree, it supports level of detail rendering. Our experiments also demonstrate its outstanding performance by interactively manipulating a data set of 3 million points such as rotating, translating and scaling with about 60 FPS.



points(m) 1
PointSize 1
opacity 1
SizeType Fixed
Materials RGB
quality Squares
Clip Mode Highlight Inside
skybox □
stats □
BoundingBox □

Figure 2: The user interface of potree.

4 EXPERIMENTAL RESULTS

We launch a Spark cluster with one master node and ten slave nodes with help of Amazon EC2 service (Amazon, 2015). For each node, m4.large instance is used and the operation system is Ubuntu 14.04. All our experiments are performed using Apache Spark 1.4.0.

The training data acquired by mobile mapping system is from (Vallet et al., 2015) and is manually annotated correctly. The ratio between the numbers of tree crown points and others in the original annotated data from (Vallet et al., 2015) is adjusted to 1:1 in order to prevent generating a biased learning model. The testing data is mobile mapping data of street scenes in Toulouse. 100 different point clouds are used in our experiments and each one contains 3 million points. Figure 4 displays the visualized results of six point clouds. As shown in the figure, the results are fairly promising – most of tree crown points are labeled out from complex street scenes consisting of various objects such as buildings, vehicles, pedestrians, sign boards and so forth.

We also explore the scalability of our implementation by executing the same test using different data sets of varying sizes as shown in Figure 3. The x axis represents the number of point clouds used for testing and each point cloud contains 3 million points. The y axis represents the running time of the experiments on different data sets. The first two times have only a 7 seconds difference due to the small data sizes – most of time is caused by system overhead. The latter three times express an approximate linear increase depending on the data sizes. Therefore, theoretically the growing data size can be offset by increasing the number of cluster nodes.

5 CONCLUSION

In this paper, a tree crown classification method is proposed and implemented in cloud platform. Apache Spark is adopted to fulfill the parallel computing. The experimental results demonstrate its promising performance for large-scale point cloud data processing.

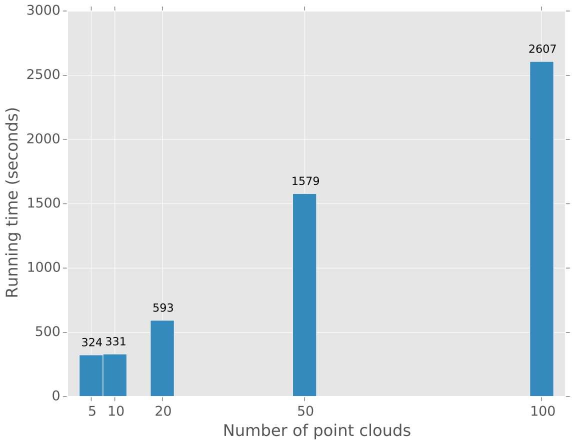


Figure 3

ACKNOWLEDGEMENTS

The authors would like to thank IGN to provide the mobile mapping data. This research work is supported by EU grant FP7-ICT-2011-318787 (IQmulus) and Amazon AWS grant.

References

AHN, 2014. AHN2, http://www.ahn.nl/pagina/open-data.html.

Amazon, 2015. Amazon EC2, http://aws.amazon.com/ec2/.

Bishop, C. M., 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Breiman, L., 2001. Random forests. Machine Learning 45(1), pp. 5–32.

Dean, J. and Ghemawat, S., 2008. Mapreduce: Simplified data processing on large clusters. Commun. ACM 51(1), pp. 107–113.

Demantké, J., Mallet, C., David, N. and Vallet, B., 2011. Dimensionality based scale selection in 3d lidar point clouds. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 38(Part 5), pp. W12.

Microsoft, 2014. Kinect, https://www.microsoft.com/en-us/kinectforwindows/.

Pix4D, 2014. Pix4Dmapper, https://pix4d.com/products/.

Potree, 2015. Potree, https://github.com/potree/potree.

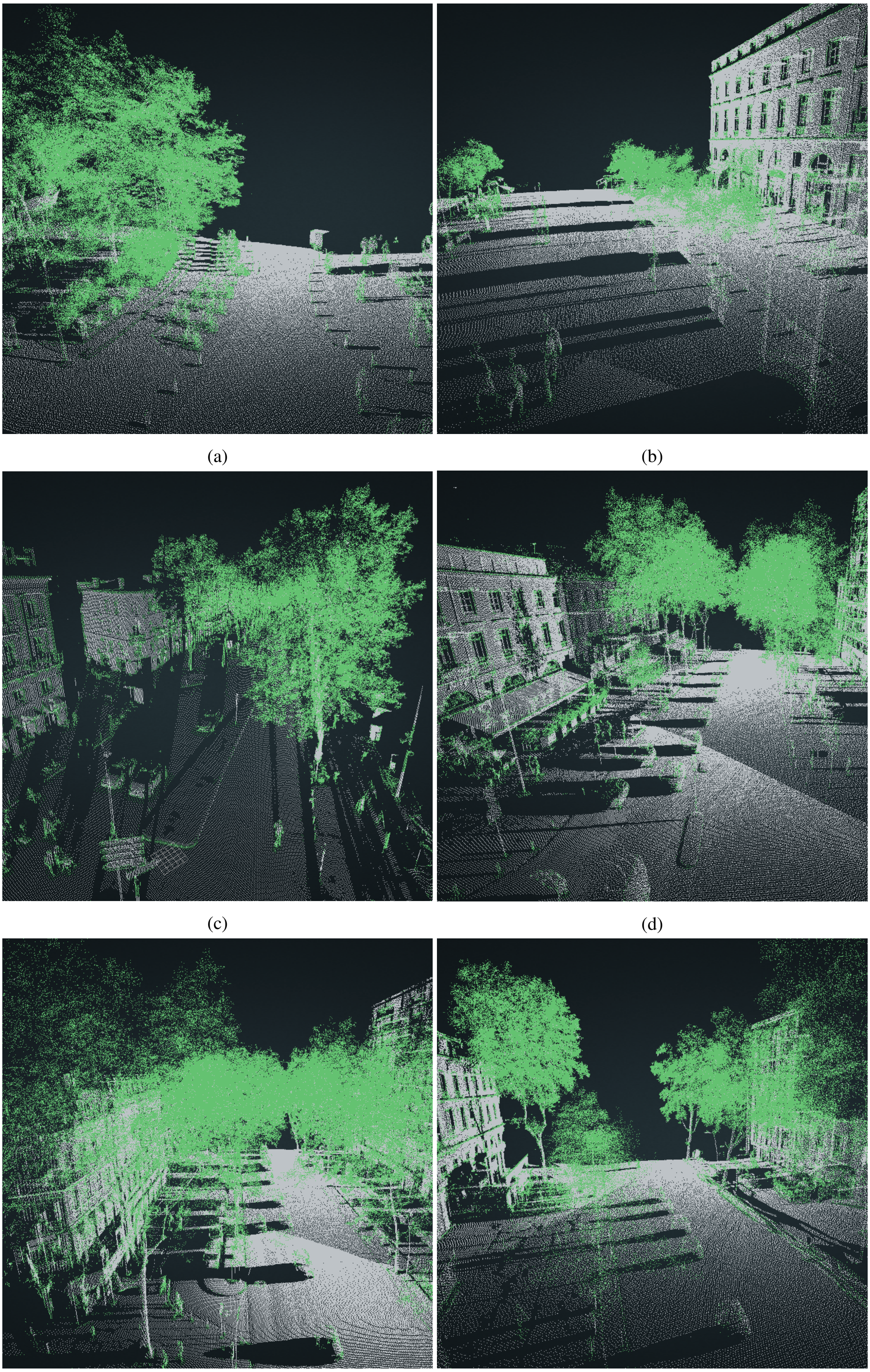
Vallet, B., Brdif, M., Serna, A., Marcotegui, B. and Paparoditis, N., 2015. Terramobilita/iqmulus urban point cloud analysis benchmark. Computers & Graphics.

Viametris, 2014. Viametris iMMS, http://viametris.info/iMMS/EN/.

Weinmann, M., Jutzi, B. and Mallet, C., 2014. Semantic 3d scene interpretation: a framework combining optimal neighborhood size selection with relevant features. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences,Volume II-3.

White, T., 2009. Hadoop: The Definitive Guide. 1st edn, O'Reilly Media, Inc.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S. and Stoica, I., 2010. Spark: Cluster computing with working sets. In: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'10, USENIX Association, Berkeley, CA, USA, pp. 10–10.



(a) (b) (c) (d) (e) (f)

Figure 4: The classification results of six point clouds acquired by mobile mapping system.