A Distributed System for Optimal Scale Feature Extraction and Semantic Classification of Large-scale Airborne LiDAR Point Clouds

Satendra Singh, and Jaya Sreevalsan-Nair

Abstract

Airborne LiDAR (Light Detection and Ranging) or aerial laser scanning (ALS) technology can capture large-scale point cloud data, which represents the topography of large regions. The raw point clouds need to be managed and processed at scale for exploration and contextual understanding of the topographical data. One of the key processing steps is feature extraction from pointwise local geometric descriptors for object-based classification. The state of the art involves finding an optimal scale for computing the descriptors, determined using descriptors across multiple scales, which becomes computationally intensive in the case of big data. Hence, we propose the use of a widely used big data analytics framework integration of Apache Spark and Cassandra, for extracting features at optimal scale, semantic classification using a random forest classifier, and interactive visualization. The visualization involves real-time updates to the selection of regions of interest, and display of feature vectors upon a change in the computation of descriptors. We show the efficacy of our proposed application through our results in the DALES aerial LiDAR point cloud.

Index Terms

Big Data framework, Apache Spark, Cassandra, LiDAR point cloud, Multiscale feature extraction, Semantic classification

I. Introduction

Three-dimensional (3D) topographical data for large expanses of region is captured effectively using airborne Light Detection and Ranging (LiDAR) technology. The data is procured in the format of point clouds, which are unstructured. Contextual understanding of such data is necessary to make sense of the environment and its constituents. Hence, semantic classification is a key processing method applied on the point clouds. The state of the art in semantic classification of LiDAR point clouds is mostly supervised learning including ensemble learning techniques, such as random forest classifiers [1], and deep learning techniques [2]. The feature vector for the learning task is obtained using local geometric descriptors computed using local neighborhood, which is determined at multiple scales [3], [1]. The combination of feature extraction at optimal scale of local neighborhood and a random forest classifier has been recommended as an appropriate framework for semantic classification in terms of both accuracy and computational efficiency [1]. Upon evaluating multiple scales, the optimal scale is determined at the *argmin* of Shannon entropy computed from eigenvalues of the covariance matrix representing the local neighborhood at the scale.

However, the existing solutions for this compute-intensive combination do not directly scale for large-scale point clouds in data analytic applications, such as, interactive feature visualization and semantic classification. As an example, DALES (Dayton Annotated LiDAR Earth Scan) [2] is one of the largest publicly available annotated point cloud dataset acquired using aerial laser scanning, with $\sim$0.5 billion 3D points at considerably high point resolution of 50 ppm (points per meter). The dataset spans a region of 10 km^2 in the city of Surrey in British Columbia, stored in 40 tiles, with each tile containing 12 million points. That said, existing big data tools and frameworks can be tapped into and re-purposed for addressing this gap. In our previous work, we have proposed an integrated framework of Apache Spark and Cassandra to perform semantic classification using feature extraction from local geometric descriptors using multiscale aggregation of saliency map [4]. Here, we extend the framework for optimal scale feature extraction, and interactive visualization (Figure 1. Our main contribution is in extending framework for semantic classification of large-scale point cloud using feature extraction at optimal scale and conventional classifiers, such as random forest classifier.

Local neighbor search is one of the most compute-intensive steps in feature extraction for point cloud processing. The computational requirements multiply when performing feature extraction across multiple scales by determining an optimal scale based on Shannon entropy [3]. Parallel processing has been exploited for implementing semantic classification has been implemented on large-scale point clouds. The classification of Semantic3D has been done using random forest classifiers using OpenMP [5], and deep learning frameworks, such as Torch [6]. k-nearest neighborhood has been used widely used type of local

This document is a preprint, as on October 10, 2020. The authors are with Graphics-Visualization-Computing Lab (GVCL), International Institute of Information Technology, Bangalore, 26/C Electronics City, Hosur Road, Bangalore 560100, India. *Corresponding author: Jaya Sreevalsan-Nair.* e-mail: jnair@iiitb.ac.in | {satendra.singh}@iiitb.org

This work was supported by the Early Career Research Award from Science and Engineering Research Board, awarded by the Department of Science and Technology, Government of India.

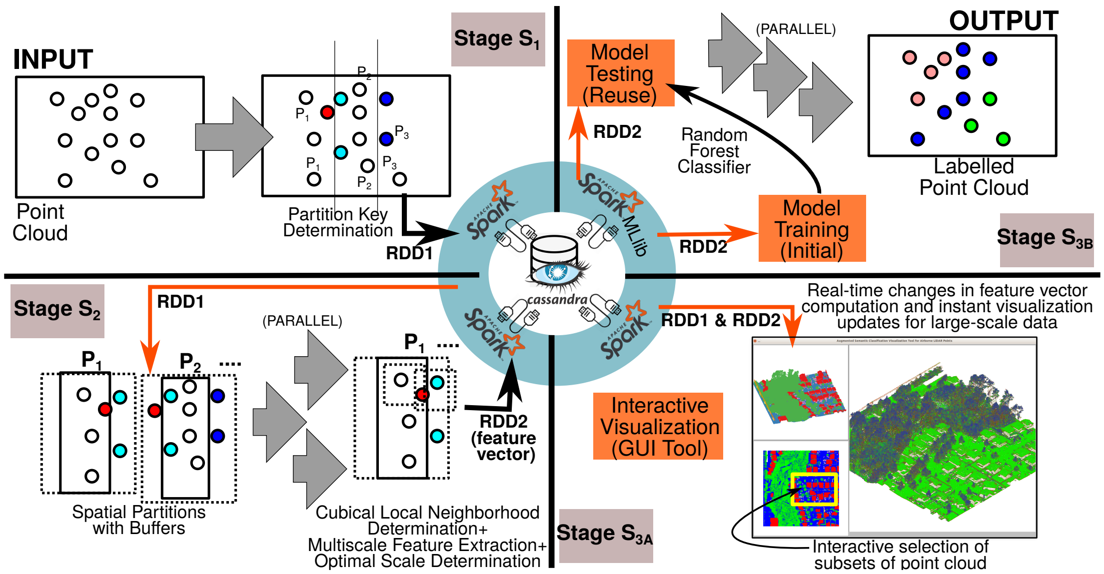

Fig. 1: Our proposed 3-stage workflow for data management, optimal scale feature extraction, and visual analytics involving interactive visual exploration and semantic classification of large-scale airborne LiDAR point clouds using Apache Spark-Cassandra integrated framework. The processed data is managed using resilient distributed datasets (RDDs), and the framework integration uses Datastax Spark-Cassandra-Connector.

neighborhood with deep learning methods to optimize the performance of these architectures. For instance, Adam optimizer has been used in RandLA-Net [7], which also performs down-sampling of the point cloud on the GPU. While parallelized and optimized machine learning frameworks can improve computational efficiency, the big data frameworks have been largely used for both dataset management and processing. Apache Spark has been used for extraction of tree crowns from LiDAR point cloud, using spherical neighborhood [8].

A. Background

Apache Spark is a unified data analytics engine for large-scale data using in-memory processing [9]. Integration of Spark with storage systems, such as key-value stores, e.g., Cassandra [10], provides persistent storage. Both Spark and Cassandra are horizontally scalable, as more distributed systems, or *nodes*, can be added to the cluster. Spark uses Resilient Distributed Data (RDD) distributed across multiple cluster nodes for loading data in logical partitions across many servers for parallel processing on the cluster nodes. Apache Spark-Cassandra Connector from Datastax [1] is used to query Cassandra tables from RDDs, after which the query results are stored in Cassandra. The connector leverages data locality to mitigate the network latency. Apache Spark also integrates complex tools such as MLlib, for machine learning.

II. Methodology

We use an integrated Apache Spark-Cassandra framework for semantic classification of large-scale airborne LiDAR point clouds using optimal scale feature extraction, and interactive visualization. We design an appropriate 3-stage workflow for utilizing this framework effectively. Here, the choice of Apache Spark with Cassandra has been made for: (a) parallelizing and scaling with data as well as nodes, and (b) optimized performance in semantic classification using tools like Spark ML, and interactive visualization. The persistent storage using Cassandra serves two purposes in our case: (a) storage of processed data in compute-intensive interactive applications, e.g., visualization, (b) distributed data management, as, in a multi-partitioned node, only a single partition can be in-memory in Spark at any given time. A partition key is needed for partitioning data across nodes, and is computed based on the user-defined strategy on Spark. A hash value of the partition key is needed for inserting and retrieving data and it is computed using a function Partitioner in Apache Spark during the read-write operations on the cluster. We exploit the feature in Cassandra to store the data in distributed nodes based on the partition key and optimize the search using *clustering columns*. In our work we partition 3D data in the x-y plane, assign the partitions a region-ID, use the region-ID as the partition Key, and assign the x, y, z variables as the clustering columns. When the Spark executor and Cassandra nodes are deployed on the same machine, the integrated framework processes the region data without incurring any network traffic, owing to the use of Spark-Cassandra Connector.

[1] Spark Cassandra Connector, https://github.com/datastax/spark-cassandra-connector

A. Our Proposed Workflow

Our workflow implemented on the integrated framework comprises of the following three stages (§Figure 1): the partition assignment of large-scale point cloud on the framework [$\mathbb{S}_1$], spatial partitioning and feature extraction [$\mathbb{S}_2$], and visual analytics [$\mathbb{S}_3$]. Within $\mathbb{S}_3$, the framework performs interactive visualization of features in the point cloud, $\mathbb{S}_{3A}$, and semantic classification, $\mathbb{S}_{3B}$. Compared to our previous work [4], here, we compute more features, modify $\mathbb{S}_2$ to determine the optimal scale, and include $\mathbb{S}_{3A}$.

Stage $\mathbb{S}_1$: For initializing the framework, we load the 3D point cloud $\mathscr{P}$ into the Apache Spark as an RDD. We normalize all points in $\mathscr{P}$ to be contained inside a cube of size 2 units centered at (0,0,0), without altering its aspect ratio. We then partition only along one axis, referred to as the *principal axis*, to strategize the partition layout with fewer partition boundaries. The partition boundaries pose an overhead of inter-node communication, as, for the points close to the boundaries, their local neighborhoods are split across different partitions, and hence, across different nodes. Thus, fewer partition boundaries are used to reduce the inter-node communication for collating neighborhood information. We select the axis with maximum range as principal axis p, either x or y axes, here. Spatial partitions of $\mathscr{P}$ into N contiguous regions along the p axis, have partition boundaries at p_i, for $i = 0, 1, 2, \ldots, N$, where N is determined using the maximum scale value, l_{max}, range of data along p-axis in $\mathscr{P}$ ($\Delta p = p_{max} - p_{min}$), and the number of available cluster nodes n. Thus, $N = \frac{\Delta p}{l_{max}.n}$, and the i^{th} partition boundary $p_i = p_{min} + i * n * l_{max}$. A region-ID assigned to each point x in $\mathscr{P}$, serves as the partition key in Spark, K, where the p-coordinate of the point satisfies the boundary condition, $p_{(K-1)} \leq x_p < p_{(K)}$ for $K = 1, 2, \ldots, N$. For each partition, a buffer region is introduced by extending the right and left boundaries to $p_i \pm l_{max}$, respectively. Buffer regions provide *complete* local neighborhood information for boundary points, and features are extracted for all points except those in buffer regions. We store the resultant RDD in the Cassandra cluster using partition key, K.

Stage $\mathbb{S}_2$: We create partitions with the assigned K in $\mathbb{S}_1$ using our custom partitioner in the RDD. The custom partitioner enforces our computed partitioning, thus, overriding the default random one on Spark. The partition key is crucial for the spatial contiguity in $\mathscr{P}$ as it ensures that a partition is contained in a single node without being split across nodes. A single node can still load multiple partitions, and process them in parallel. The feature extraction algorithm consists of four sequential processes implemented for each point, namely, local neighborhood determination, descriptor computation, its eigenvalue decomposition, and feature vector computation. Point-wise processing makes the algorithm embarrassingly data-parallel. Here, we use the cubical neighborhood [11] over the conventional spherical or k-nearest, to reduce computations. Cubical neighborhood uses Chebyshev distance (infinity (L_∞) or maximum norm) for neighbor-search, instead of Euclidean distance (L_2 norm). A spherical neighborhood of radius r is approximated by the cubical neighborhood of $l = 2r$.

Definition II.1. *l-cubical neighborhood* N_l of a point x in $\mathscr{P}$, such that $\mathscr{P} = \{p \in \mathbb{R}^d\}$, is a set of points which satisfy the criterion based on Chebyshev distance,

$$N_l(x) = \{y \in \mathscr{P} \mid \max_{\{0 \leq i < d\}} (|x_i - y_i|)\}.$$

The local geometric descriptor provides the shape of the local neighborhood, e.g. the covariance tensor $T_{3\mathrm{DCM}}$ [12], and its size is the *scale*. The eigenvalue decomposition of the descriptor gives the likelihood of the corresponding point being on a surface, line, or junction (point) type feature [4], given by the saliency map $[C_l, C_s, C_p]$. For eigen values of the descriptor, such that, $\lambda_1 \geq \lambda_2 \geq \lambda_3$:
C_l=$(\lambda_1 - \lambda_2)/\sum_\lambda$, C_s=$2(\lambda_2 - \lambda_3)/\sum_\lambda$, C_p=$3(\lambda_3)/\sum_\lambda$; for $\sum_\lambda$=$(\lambda_1 + \lambda_2 + \lambda_3)$.

There are different 3D features computed using geometric and the shape properties [1]. The eight different local 3D shape features using eigenvalues of the descriptors are: omnivariance O_λ, eigen-sum $\sum_\lambda$, eigen-entropy E_λ, change of curvature C_λ, linearity L_λ, planarity P_λ, scattering S_λ, and anisotropy A_λ. O_λ and $\sum_\lambda$ are tensor invariants of second-order tensor, namely, determinant and trace; with O_λ=$\sqrt[3]{\lambda_1\lambda_2\lambda_3}$. Eigen-entropy gives the Shannon entropy in descriptor shape, given by E_λ=$-\sum_{i=1}^{3} \lambda_i \ln(\lambda_i)$. Other measures are: C_λ=$\lambda_3/\sum_\lambda$, L_λ=$(\lambda_1 - \lambda_2)/\lambda_1$, P_λ=$(\lambda_2 - \lambda_3)/\lambda_1$, S_λ=λ_3/λ_1, A_λ=$(\lambda_1 - \lambda_3)/\lambda_1$. Since C_p and C_λ are equivalent, we ignore C_λ. The semantics of the saliency map $[C_l, C_s, C_p]$ and the descriptor shape $[L_\lambda, P_\lambda, S_\lambda]$ are the same; thus, we keep $[C_l, C_s, C_p]$. Of four geometric features we use, three are height-based, namely, the absolute height z of each point, and the range z_Δ and standard deviation z_σ of the height distribution in the local neighborhood of the point. The fourth feature is local point density D [1], given by $D = (n_p + 1)/(\frac{4}{3}\pi r^3)$, where n_p is the number of points in the l-cubical neighborhood, and $r = 0.5l$. Thus, we get a 11-feature vector at each point in $\mathscr{P}$ as:

$$\mathbf{v}_f = [z, z_\Delta, z_\sigma, D, C_l, C_s, C_p, O_\lambda, \textstyle\sum_\lambda, E_\lambda, A_\lambda].$$

In the case of annotated data, the class label for each point is stored along with $\mathbf{v}_f$ in an RDD in Spark and the Cassandra cluster, using K. The class label is used for training data for the classifier, and validation.

Optimal scale determination: We compute the feature vector at different scales, i.e. size of the cubical neighborhood, l, such that $l_{min} \leq l \leq l_{max}$, using n_s uniform scales. Thus, scale step-size is Δl=$\frac{(l_{max}-l_{min})}{(n_s-1)}$. The optimal scale is the *argmin* of E_λ. We then use the feature vector $\mathbf{v}_f^{(i)}$ at the optimal scale at each point as the feature vector for the concerned point in the classification stage ($\mathbb{S}_{\mathbf{3B}}$). These point-wise feature vectors at different scales are stored in the same RDD to determine the global minimum of E_λ [3], and thus, the optimal scale.

Stage $\mathbb{S}_{\mathbf{3A}}$: Our visualization tool is inspired by Potree [13], a browser-based visualization tool for large-scale point clouds using WebGL. It loads the data from file, organizes data in the octree data structure, and stores on the local disk on the web server. Potree provides interactive visualization of the point clouds loaded from file, using Poisson disk sampling. Potree, however, does not perform real-time analytics. To facilitate real-time analytics and visualization, we load the Cassandra-resident data on the local disk as required, and render the point clouds using OpenGL on a desktop application. We propose the system architecture to perform not just interactive visualization but also perform selective analytics on the fly using Apache Spark. The real-time performance is facilitated by Cassandra storage, and the Spark-Cassandra Connector. As an example usage of our visualization tool, we change scale on the fly and visualize the features computed for the scale, using parallel processing.

Stage $\mathbb{S}_{\mathbf{3B}}$: For training, the feature RDD of the training data is loaded into Apache Spark ML. Then, any classifier on Spark ML, e.g. random forest classifier (RFC) or gradient boost tree classifer (GBT), is initiated, and stored as a classifier model in file. For testing the model, the feature RDD of the testing data is loaded, and the classifier is run on $\mathbf{v}_f$ to determine point-wise class labels. The resultant RDD with the $\mathbf{v}_f$ and the class label is stored in the local Cassandra node, and efficiently retrieved using the Connector. We perform training/testing using 80/20 split, and for labelled data, we validate the model seamlessly.

III. Experiments and Results

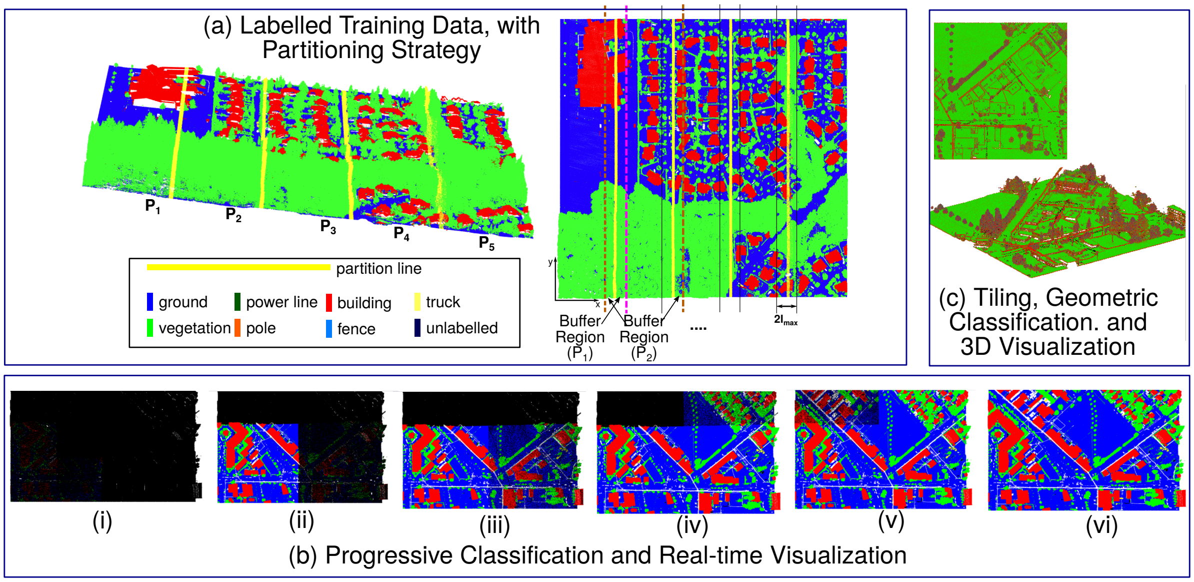

Fig. 2: Visualization of a region 5110_54495 of DALES dataset (∼12 million points) – (a) Labeled data for training, and partitioning strategy for reduced boundaries; (b) real-time updates of semantic classification of airborne LiDAR point clouds in our visualization system interfacing with our distributed system, from (i) to (vi), and its real-time updates from unlabeled points (black) to class labels; (c) tiling of the point cloud, computation of saliency map on the fly, and point rendering with color (RGB) corresponding the saliency map (C_l, C_s, C_p), with interactive rotation of the section.

We have used Apache Spark 2.4 and Apache Spark ML, integrated with Cassandra 3.0., with three executor nodes on Apache Spark, of which one executor node runs on master node. All the four nodes use Intel i7 processor @2.80 GHz, 4 cores, 8 logical processors, and 8GB RAM. For our experiment, we have used the Dayton Annotated LiDAR Earth Scan (DALES) dataset [2], which is one of the largest aerial LiDAR point clouds (Figure 2), with 0.5 billion points across 8 semantic classes, stored in 40 tiles. In our distributed system, there are five spatially contiguous partitions (Figure 2), of which one partition is loaded on the executor running on master node, and two each in the other executor nodes. We have used feature vectors computed at optimal scale determined from 10 scales, with cubical neighborhood sizes l=$\{1m, 2m, 3m, 4m, 5m, 6m, 7m, 8m, 9m, 10m\}$ in the normalized coordinates. We have trained the RFC on Spark ML using ∼34 million points in tiles 5110_54460, 5110_54475, and 5110_54495 of DALES (Figure 2), and tested on ∼11 million points in another tile of DALES, 5080_54470.

In our case study, we observe that the buffer regions add data for $10m$, on either side of each partition. This implies each partition has up to $8K$ points (∼1.3MB) overhead, with 50 ppm. This becomes a significant overhead when we consider massive point cloud datasets, as the buffer region grow proportional to the point cloud size. However, when we take a closer look at the feature vector, we observe that the local descriptor used for generating a predominant part of the vector is an additive tensor. Hence, when the local neighborhood is truncated for the boundary points in a partition, the local descriptor is a coarser

TABLE I: Semantic classification result for our case study of DALES point cloud, using 33,825,345 training points, using different classifiers in a distributed system

Buffer Region (# test points)	OA	mean	ground	vegeta -tion	cars	trucks	power line	fence	pole	building
Random Forest Classifier (RFC)										
With (10,773,000)	0.817	0.357	0.781	0.739	0.154	0.199	0.238	0.159	0.190	0.395
Without (12,654,558)	0.798	0.327	0.760	0.703	0.155	0.186	0.153	0.134	0.182	0.346
Gradient Boost Tree (GBT) Classifier										
With (10,642,978)	0.792	0.351	0.746	0.626	0.030	0.110	0.447	0.177	0.206	0.464
Without (12,654,558)	0.773	0.341	0.719	0.657	0.041	0.133	0.487	0.217	0.149	0.321

approximation. The overall accuracy (OA) of semantic classification is not expected to be affected drastically owing to the low percentage of boundary points. As an experiment, we estimate the trade-off.

We have determined the Intersection Over Union (IoU) values for mean, overall accuracy (OA), and per class (Table I). RFC gives an OA of 81.7%, with 78.1% for ground class. We observe that the absence of buffer region gives us an OA of 79.8% with 76% for ground class. For each square tile of $0.5km$, the total buffer region with 5 partitions is $0.1km$. Thus, we observe that we can have a trade-off of 16% of additional storage by 2% reduction in overall accuracy in semantic classification. RFC and GBT classifier perform comparably. We demonstrate the use of our visualization system in progressive rendering of the progression in semantic classification as the results get updated; and the visualization of saliency map in the point cloud, on the fly after sectioning (Figure 2).

IV. Conclusions

In this paper, we have explored the use of an integrated Apache Spark-Cassandra framework as a distributed system for optimal scale feature extraction from a point cloud and its semantic classification, using customized region-based spatial partitioning. Our proposed partitioning includes buffer regions for including the local neighborhood of the partition boundary points. Overall, our proposed 3-stage workflow for interactive visualization and semantic classification has been effectively implemented on the integrated framework using these design choices. We have observed that not using buffer region saves the additional 16% of storage needed, with only 2% reduction in overall accuracy of classification. We are currently improving classification results on our system and exploring larger datasets on it, over a wide range of other applications in point cloud management and processing.

References

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

[2] Nina Varney, Vijayan K Asari, and Quinn Graehling, "DALES: A Large-scale Aerial LiDAR Data Set for Semantic Segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 186–187.

[3] Jérôme Demantké, Clément Mallet, Nicolas David, and Bruno Vallet, "Dimensionality based Scale Selection in 3D LiDAR Point Clouds," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 38, no. Part 5, pp. W12, 2011.

[4] S. Singh and J. Sreevalsan-Nair, "A Distributed System for Multiscale Feature Extraction and Semantic Classification of Large-scale LiDAR Point Clouds," in *Proceedings of the IEEE India Geosc. & Remote Sensing Symp. (to appear)*, 2020.

[5] Timo Hackel, Jan D. Wegner, and Konrad Schindler, "Joint Classification and Contour Extraction of Large 3D Point Clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 130, pp. 231–245, 2017.

[6] Timo Hackel, Jan D. Wegner, Nikolay Savinov, Lubor Ladicky, Konrad Schindler, and Marc Pollefeys, "Large-scale supervised learning For 3D point cloud labeling: Semantic3d. Net," *Photogrammetric Engineering & Remote Sensing*, vol. 84, no. 5, pp. 297–308, 2018.

[7] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham, "RandLA-Net: Efficient semantic segmentation of large-scale point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11108–11117.

[8] Kun Liu and Jan Boehm, "Classification of Big Point Cloud Data Using Cloud Computing," *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 40, pp. 553–557, 2015.

[9] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al., "Apache spark: a unified engine for big data processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[10] Avinash Lakshman and Prashant Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, 2010.

[11] Kenneth Olofsson and Johan Holmgren, "Single Tree Stem Profile Detection Using Terrestrial Laser Scanner Data, Flatness Saliency Features and Curvature Properties," *Forests*, vol. 7, no. 9, pp. 207, 2016.

[12] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle, "Surface Reconstruction from Unorganized Points," *Siggraph Comp. Graph.*, vol. 26, no. 2, pp. 71–78, 1992.

[13] Markus Schütz, "Potree: Rendering large point clouds in web browsers," *Technische Universität Wien, Wiedeń*, 2016.