Computers & Geosciences 102 (2017) 139–147

Contents lists available at ScienceDirect

Computers & Geosciences

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

Research paper

A scalable approach for tree segmentation within small-footprint airborne LiDAR data

Hamid Hamraz[a,*], Marco A. Contreras[b], Jun Zhang[a]

[a] Department of Computer Science, University of Kentucky, Lexington, KY 40506, USA
[b] Department of Forestry, University of Kentucky, Lexington, KY 40506, USA

ARTICLE INFO

Keywords:
Distributed computing
Big spatial data
Remote sensing
Remote forest inventory
Individual tree information

ABSTRACT

This paper presents a distributed approach that scales up to segment tree crowns within a LiDAR point cloud representing an arbitrarily large forested area. The approach uses a single-processor tree segmentation algorithm as a building block in order to process the data delivered in the shape of tiles in parallel. The distributed processing is performed in a master-slave manner, in which the master maintains the global map of the tiles and coordinates the slaves that segment tree crowns within and across the boundaries of the tiles. A minimal bias was introduced to the number of detected trees because of trees lying across the tile boundaries, which was quantified and adjusted for. Theoretical and experimental analyses of the runtime of the approach revealed a near linear speedup. The estimated number of trees categorized by crown class and the associated error margins as well as the height distribution of the detected trees aligned well with field estimations, verifying that the distributed approach works correctly. The approach enables providing information of individual tree locations and point cloud segments for a forest-level area in a timely manner, which can be used to create detailed remotely sensed forest inventories. Although the approach was presented for tree segmentation within LiDAR point clouds, the idea can also be generalized to scale up processing other big spatial datasets.

1. Introduction

Individual tree information is increasingly becoming the preferred data precision level to accurately and efficiently monitor, assess, and manage forest and natural resources (Chen et al., 2006; Koch et al., 2006; Schardt et al., 2002). In the last two decades, airborne light detection and ranging (LiDAR) technology has brought drastic changes to forest data acquisition and management by providing inventory data at unprecedented spatial and temporal resolutions (Ackermann, 1999; Maltamo et al., 2014; Shao and Reynolds, 2006; Swatantran et al., 2016; Wehr and Lohr, 1999). However, to obtain accurate tree level attributes such as crown width and tree height as well as derivative estimates such as diameter at breast height (DBH), volume, and biomass, accurate and automated tree segmentation approaches are required (Schardt et al., 2002).

Numerous methods for tree segmentation within LiDAR data have been proposed (Duncanson et al., 2014; Hamraz et al., 2016; Hu et al., 2014; Jing et al., 2012; Li et al., 2012; Persson et al., 2002; Popescu and Wynne, 2004; Véga and Durrieu, 2011; Véga et al., 2014; Wang et al., 2008). Nevertheless, these methods have only been experimented for small forested areas and none of them have thoroughly considered scalability; LiDAR data covering an entire forest is much more voluminous than the memory of a typical workstation and may also take an unacceptably long time to be sequentially processed. Also, given the continuous advancements of the sensor technology (Swatantran et al., 2016), the LiDAR point clouds will be acquired with less costs and greater resolutions, which in turn increases the need for more efficient and scalable processing schemes.

A few studies have considered processing LiDAR data (Thiemann et al., 2013; Zhou and Neumann, 2009) using streaming algorithms (Pajarola, 2005), where the spatial locality of the LiDAR data is used to construct out-of-core algorithms. However, streaming algorithms are unable to reduce the time required for processing because of their inherently sequential processing scheme. A number of recent studies have considered leveraging the power of multicore and/or GPU (shared memory) platforms for processing LiDAR data for efficient DEM modeling (Guan and Wu, 2010; Oryspayev et al., 2012; Sten et al., 2016; Wu et al., 2011), or for 3D visualization (Bernardin et al., 2011; Li et al., 2013; Mateo Lázaro et al., 2014), although shared-memory platforms are also bounded in the amount of memory and the number of processing units.

On the other hand, processing geospatial data such as LiDAR data

* Corresponding author.
E-mail addresses: hhamraz@cs.uky.edu (H. Hamraz), marco.contreras@uky.edu (M.A. Contreras), jzhang@cs.uky.edu (J. Zhang).

http://dx.doi.org/10.1016/j.cageo.2017.02.017
Received 3 October 2016; Received in revised form 30 November 2016; Accepted 19 February 2017
Available online 22 February 2017
0098-3004/

can be parallelized by partitioning the data into tiles (commonly used for data delivery purposes) and distributing the tiles to different processors on a distributed architecture. Huang et al. (2011) proposed a master-slave distributed method for parallelizing inverse distance weighting interpolation algorithm. Guan et al. (2013) designed a cloud -based process virtualization platform to process vast quantities of LiDAR data. Barnes (2016) parallelized Priority-Flood depression-filling algorithm by subdividing a DEM into tiles. However, the above distributed approaches were designed and used for perfectly parallel problems while, in case of non-perfectly parallel problems, dealing with the data near the boundaries of the tiles is not trivial and should be elaborated according to the specifics of the application (Werder and Krüger, 2009).

Accounting for the data near the tile boundaries, a distributed density-based clustering for spatial data (Ester et al., 1996) was presented by Xu et al. (2002). The authors proposed a master-slave scheme in which the master spawns a number of slaves to perform the clustering and return the result back to the master, who then combines the results. The scheme relies on a data placement strategy for load balancing in which the master partitions the data and distributes the portions among the slaves for processing, hence the runtime is determined by the last slave that finishes its job. Distributing the data and merging the results by the master are also sequential procedures and may yield performance bottlenecks. A more recent work (He et al., 2011) has presented a version of the density-based clustering tailored to run on a MapReduce infrastructure (Dean and Ghemawat, 2008) performing four stages of MapReduce for indexing, clustering, as well as identifying and merging boundary data. The MapReduce infrastructure, although constraining the programming model, has the advantage of built-in simplicity, scalability, and fault tolerance. Thiemann et al. (2013) have presented a framework for distributed processing of geospatial data, where partitioning the data to tiles with overlapping areas near the borders is their core solution. The overlapping area should be at least as big as the required neighborhood for processing a local entity and the produced overlapping result may require special treatment to be unified. The authors used the map phase of the Hadoop MapReduce infrastructure (White, 2012) for clustering buildings of large urban areas and the overlapping result was unified separately afterwards.

Although there are various methods proposed for tree segmentation, only few studies have considered scalable processing of large geospatial data – there is specifically no study considering forest-level datasets. This is increasingly important when obtaining tree-level information for areas other than small-scale plots, which is often the case when obtaining LiDAR data. This paper presents and analyzes a distributed approach that accounts for the data near the tile boundaries and uses a tree segmentation algorithm as a building block in order to efficiently segment trees from LiDAR point clouds representing an entire forest. For experimentation, the approach was implemented using message passing interface (MPI) (Walker, 1994).

2. Materials and methods

2.1. LiDAR data

We used LiDAR data acquired over the University of Kentucky Robinson Forest (Lat. 37.4611, Long. −83.1555), which covers an aggregated area of 7441.5 ha in the rugged eastern section of the Cumberland Plateau region of southeastern Kentucky in Breathitt, Perry, and Knott counties (37°28′23″N 83°08′36″W) (Overstreet, 1984). The LiDAR data is a combination of two datasets collected with the same LiDAR system (Leica ALS60 at 200 kHz flown with an average speed of 105 knots) by the same vendor. One dataset was low density (~1.5 pt/m^2) collected in the spring of 2013 during leaf-off season (average altitude of 3096 m above the ground). The second dataset was high density (~25 pt/m^2) collected in the summer of 2013

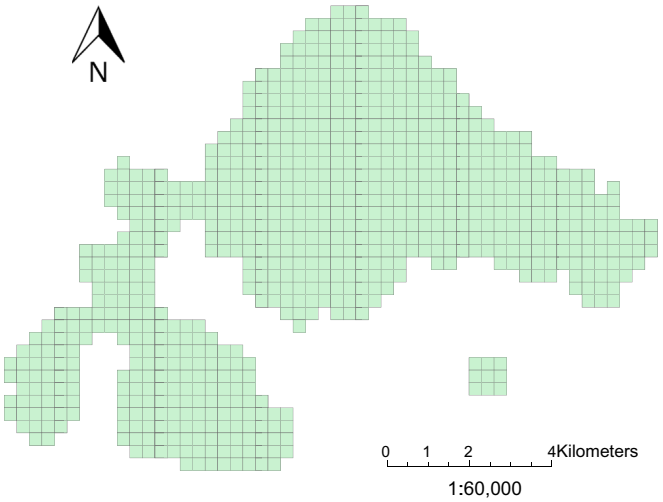

Fig. 1. LiDAR tile map of Robinson Forest consisting of 801 9.3-ha tiles.

during leaf-on season (average altitude of 196 m above the ground). The combined dataset has a nominal pulse spacing (NPS) of 0.2 m and was delivered in 801 square (304.8 m side ~ 9.3 ha area) tiles (Fig. 1), each containing about 5 million LiDAR points on average and occupying about 400 MB of disk space. The entire LiDAR dataset contains over 4 billion points and occupies 320 GB of disk space.

2.2. Distributed processing

In a distributed processing environment, the LiDAR data representing tree crowns located across tile boundaries is split into two or more pieces that are processed by different processing units. Identifying such crown pieces, unifying them, and efficiently managing the distributed resources to run with a reasonable speedup are the main challenges of a distributed approach. We propose a master-slave

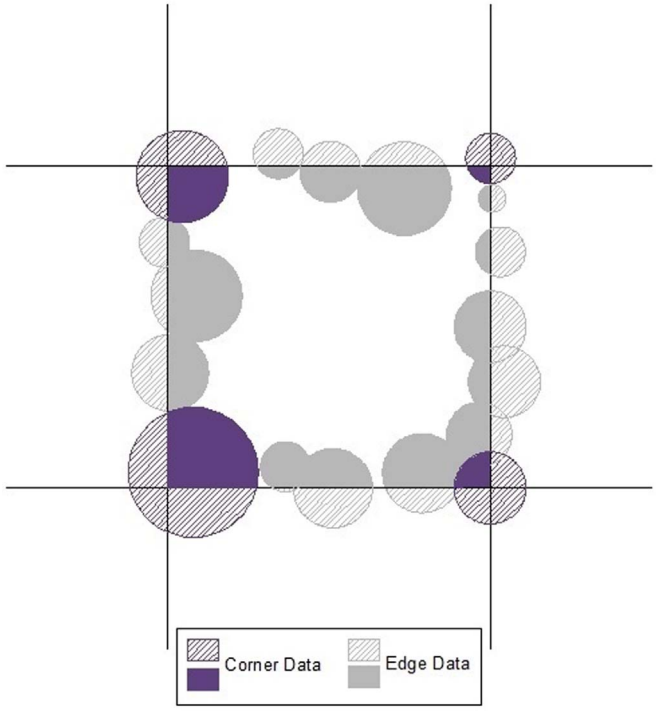

Fig. 2. A schematic of a tile with the two types of boundary data. The solid-colored tree crown pieces inside the tile should be unified with the corresponding stripe-colored parts outside.