A Vector-Based Method for Drainage Network Analysis Based on LiDAR Data

Fangzheng Lyu[1, 2], Zewei Xu[1, 2], Xinlin Ma[3], Shaohua Wang[1, 2], Zhiyu Li[1, 2],

and Shaowen Wang[1, 2, *]

[1]CyberGIS Center for Advanced Digital and Spatial Studies

[2]Department of Geography and Geographic Information Science

University of Illinois at Urbana-Champaign

[3]College of Urban and Environmental Science, Peking University

*Corresponding author: Shaowen Wang

E-mail addresses: shaowen@illinois.edu

Address: Natural History Building 2046, MC-150, 1301 W. Green St., Urbana, IL 61801

Authorship Statement:

Code: https://github.com/cybergis/Drainage-System-Analysis-Research

Highlights:

· Vector-based drainage system analysis that enables water to flow in any direction

· Drainage system analysis and streamline detection based on fine-resolution LiDAR data

· High-performance computing for resolving computational intensity of fine-scale drainage network analysis

Keyword: Drainage Network Analysis, Vector-based Model, LiDAR, Parallel Computing

Abbreviations:
Digital Elevation Model (DEM)
Light Detection and Ranging (LiDAR)
Triangular Irregular Network (TIN)
High Performance Computing (HPC)
United States Geological Survey (USGS)
Kernel Density Estimation (KDE)

Abstract

Drainage network analysis is fundamental to understanding the characteristics of surface hydrology. Based on elevation data, drainage network analysis is often used to extract key hydrological features like drainage networks and streamlines. Limited by raster-based data models, conventional drainage network algorithms typically allow water to flow in 4 or 8 directions (surrounding grids) from a raster grid. To resolve this limitation, this paper describes a new vector-based method for drainage network analysis that allows water to flow in any direction around each location. The method is enabled by rapid advances in Light Detection and Ranging (LiDAR) remote sensing and high-performance computing. The drainage network analysis is conducted using a high-density point cloud instead of Digital Elevation Models (DEMs) at coarse resolutions. Our computational experiments show that the vector-based method can better capture water flows without limiting the number of directions due to imprecise DEMs. Our case study applies the method to Rowan County watershed, North Carolina in the US. After comparing the drainage networks and streamlines detected with corresponding reference data from US Geological Survey generated from the Geonet software, we find that the new method performs well in capturing the characteristics of water flows on landscape surfaces in order to form an accurate drainage network.

1. Introduction

Drainage network analysis is fundamental to understanding the characteristics of surface hydrology and can be conducted based on elevation data to derive both stream channels and drainage networks of the underlying landscape. Rapid advances of LiDAR remote sensing and high-performance computing have increasingly enabled cutting-edge landscape and hydrology research (Rahil et al., 2016; Xu et al., 2018; Lukač and Žalik, 2013; Salach et al., 2018). In order to take advantage of the fine-resolution and high-accuracy characteristics of LiDAR data, drainage network analysis that is typically based on Digital Elevation Models (DEMs) needs to be innovated. While raster-based DEMs have widely been used in drainage system analytics, such models pose a limitation on the accuracy of the geospatial analysis. This is because even a fine-resolution DEM dataset contains only one point per square meter and the elevation represented by each data point often has limited precision. Furthermore, such DEM data is constrained by its two-dimensional (2D) raster model and the fact that many grid-based values are interpolated using ground return points of LiDAR causing unnecessary errors.

Another limitation of conventional drainage network analysis is that instead of allowing water to flow in any direction naturally, commonly used methods like D8 and D-infinity (Tarboton, 1997) constrain water only to flow from one grid to one or more adjacent grids, causing inevitable inaccuracies. Especially for large-scale DEM data, errors from these methods would accumulate as large spatial domains are treated.

This paper describes a novel vector-based approach to drainage network analysis based on discrete-return airborne LiDAR data. First, we represent an area using a collection of grid cells. For each grid cell, a vector is derived to represent its flow direction instead of having flow direction heading to one or more adjacent cells. At the end of the initially generated vectors, another set of vectors are constructed to represent the continuation of flows. This recursive process continues until reaching the boundary of a given area, and a flow direction map is generated as the final output. In this way, the flow direction of water is more accurately estimated by allowing water to flow in any direction. Finally, the streamlines are extracted by thresholding the number of vectors going through each grid cell.

The proposed method improves the accuracy of drainage networks from two perspectives. First, the derived drainage network is no longer constrained by the rigid 2D raster format as the network is directly derived from LiDAR data with high accuracy. Second, the method is conducted based on individual points to consider the free-flowing characteristics of drainage channels. By allowing water to flow to any adjacent areas instead of flowing into a limited number of surrounding grids, the method frees the traditional drainage system analysis from the shackle of theoretical simplification of the flowing process of natural water. Since our approach is computationally intensive, it is necessary to employ high-performance computing based on cyberGIS capabilities (Wang et al., 2019; Lyu et al., 2019). We support computational reproducibility by having the code and related data available online as a CyberGIS-Jupyter notebook that can access high-performance computing resources (Yin et al., 2018). In addition,

the final outcome is in the form of a vector field, which better supports the visualization of drainage networks.

2. Related Work

Two widely used methods for drainage network analysis based on DEM are D8 and D-infinity (Tarboton, 1997). In order to find contributing and dispersal areas, digital elevation model network (DEMON) (Costa-Cabral and Burges, 1994) and multiple flow direction algorithms (Freeman, 1991) are used to simulate water flow on landscape. These methods were designed for the representation of flow direction and calculation of the upslope area. In a recent study, Wang and Ai (2018) explored drainage networks with an innovative hexagonal grid-based DEM representation instead of raster DEM. Their model was proved to have better performance compared to the D8 model. Chen et al. (2014) proposed a vector-based flow path tracking algorithm using a triangular irregular network (TIN) representation to simulate flow dynamics. As LiDAR data has become widely available during recent years, new opportunities for extracting drainage network from LiDAR data have been extensively studied. There are two types of methods involving drainage system analysis based on LiDAR data: 1) using LiDAR-derived DEM; and 2) directly using LiDAR data. On the one hand, LiDAR-derived DEM outperforms traditional DEM for accurately delineating drainage network. Hopkinson et al. (2009) analyzed watershed attributes using three data sources (LiDAR, photogrammetry, public digital contour data) at two resolutions (5 m, 25 m) and found that a watershed area derived from the digital contour-derived DEM was overestimated by 15%, compared to a LiDAR-derived

DEM. A similar result was obtained by Murphy et al. (2008) after extracting drainage networks from 1-m resolution LiDAR-derived DEM and traditional photogrammetric DEM. Furthermore, Roelens et al. (2018) were able to extract a drainage network from 1-meter resolution LiDAR-derived DEM and get positional accuracy of as small as 0.4 m. In particular, LiDAR-derived DEM outperforms in the landscape where elevation change is high. Goulden et al. (2014) compare landscapes where the elevation change is high, medium, and low, and they find that places, where the elevation change is high, can be more sensitive to LiDAR-derived DEM while in places where the elevation change is low, the drainage network results from traditional DEM and LiDAR-derived DEM are similar. At the same time, in a landscape scenario where the drainage network is dense, the spatial resolution of DEM plays an important role in the generation of drainage network (ArizaVillaverde, 2015). On the other hand, there are studies that attempt to extract drainage networks directly using LiDAR data to avoid the bias from the process of transforming LiDAR data into DEM. Anderson and Ames (2011) proposed a method for extracting stream channels with LiDAR data using the D8 method. The performance of their method for extracting stream channels demonstrates the potential of using LiDAR data directly for drainage network analysis. Our method described in this article also directly extracts drainage networks from LiDAR data.

As drainage system analysis is important to understanding hydrological features of landscape, there are a variety of software implementations for drainage system analysis algorithms and models. For example, Jasiewicz and Metz (2011) developed a toolkit for Hortonian analysis of drainage network in GRASS GIS. To solve large-scale problems with fine-resolution data, high-

performance parallel computing is adopted to improve the computational performance of large-scale modelling of flow path networks (Mower, 1994; Zhang and Zhou, 2019; Wang et al., 2012). Several parallel computing approaches have been proposed to speed up drainage network analysis, including multi-platform shared memory and multiprocessing programming (Burger et al., 2014; Liu et al., 2014), message passing interface (MPI) (Zhang et al., 2012), graphic processing units (GPU) (Qin and Zhan, 2012; Steinbach and Hemmerling, 2012; Ortega and Rueda 2010; Sten et al., 2016), and cluster computing (Tesfa et al., 2011; Lu et al., 2012). Spatial data characteristics are often exploited to resolve the computational intensity of geospatial data analytics through parallel computing (Wang and Armstrong, 2009). While an essential pre-processing step for parallel computing of LiDAR data analytics often uses regular grids, this approach does not consider neighbourhood features as a vector chain. This research designs a new algorithm for capturing neighbourhood features through a vector chain representation for drainage network analysis using parallel computing.

3. Study area

As shown in Figure 1 (left), our study area is a watershed in Rowan County located in North Carolina, USA. Situated in the Central Interior and Appalachian ecological division, the study area contains a set of tributaries that flows into the Second Creek, which is the streamline feature of 12-digit NHD watershed (030401020504) (Xu et al., 2021; Comer et al., 2003). Figure 1 also shows a National Agriculture Imagery Program image (middle-left), Digital Elevation Model (middle-right), Reference Data For Streamline (right) for our study area in the Rowan County

watershed. The study area contains a watershed and there is a reference dataset of a real river network provided by US Geological Survey (USGS) for evaluating our method (Fig. 1). In this study, we focus on a portion of the river network as our study area.

The dataset is obtained from USGS including 31 tiles of a LiDAR dataset (around 21.3GB) that were collected in the Rowan County watershed where the dominant land cover types of the area are forest, wetland, and agriculture. It has an area of 18.11 square kilometers with an abrupt elevation variation (222m) across the entire area, which is desirable for evaluating drainage system analysis.

The point density of the LiDAR dataset is around 43 points per square meter, indicating the LiDAR dataset contains way more points than traditional DEM data per square meter. The projection system applied to the dataset is 2011 StatePlane North Carolina. Compared to traditional fine-resolution DEMs that contain one single point per grid cell, this LiDAR dataset contains much more detailed terrain feature information, which could make drainage network analysis more representative of the real-world situation (Fig. 2).

Figure 2 visualizes a part of the dataset. Different from traditional remote sensing data, the LiDAR point cloud data consists of a set of points with each being represented with different attributes including types, longitude, latitude and elevation. The left part of Figure 2 shows the ground return points of LiDAR collected. As we can see in this visualization, the red areas represent places where the elevation is relatively high while the blue areas represent places

where the elevation is relatively low. Once we zoom in the image, as shown in the right part of Figure 2, the LiDAR data can be seen as a huge collection of points (LiDAR point cloud) with different colors representing different types; the white space indicates no LiDAR point collected. Apart from longitude, latitude and elevation values, each one of the LiDAR data points contains a series of attributes such as intensity value, return number, scan angle, and GPS time stamp. In this study, since only the ground surface points are needed, non-ground points are filtered out.

4. **Method**

Our vector-based method analyzes drainage networks with the following five major steps: 1) construct a virtual extent and conduct spatial interpolation, 2) generate a vector to derive flow direction, 3) construct a chain of vectors, 4) track each chain of vectors, 5) threshold the dataset to extract the drainage network and major stream channels.

4.1 Coordinate System and Interpolation

First, a bounding box with the spatial extent of the landscape is generated. Since LiDAR points are irregularly distributed in 3D space, we create a bounding box with a virtual extent as a container for the LiDAR dataset to construct a 3D landscape. After creating the extent and transforming all the collected LiDAR ground return points onto the coordinate system, we further examine the elevation of all the points. The elevation value at any location can be obtained by spatial interpolation based on neighbouring LiDAR points. In this way, a ground

surface model can be constructed. In this paper, bilinear interpolation is adopted because it is commonly used to take advantage of existing neighbouring points (Polidori and Chorowicz 1993).

Different from traditional raster data based on DEM, LiDAR point cloud data consists of denser and more irregularly distributed data points. When spatial interpolation is performed to derive the elevation of a data point based on surrounding points, it is expected that there are enough existing data points near the data point of interest to obtain an accurate interpolation result. The density of the LiDAR point cloud in our dataset is around 43 points per square meter, which is supportive of achieving high accuracy of spatial interpolation. This is a key advantage of exploiting LiDAR data in our research.

4.2 Vector for flow direction

After coordinate system transformation and interpolation, each integer pair *(a, b)* representing a location point on the landscape has a vector representing its flow direction. Figure 3 shows the method to detect the flow direction at a specific point. Typically, the estimated flow direction at a given point is the direction with the steepest gradient descent.

In order to compute the direction that water flows to from point *(a, b)* in Figure 3, a circle of 1m radius (smaller radius generates higher-resolution drainage network results but with higher computational intensity) is generated around the point *(a, b)* as it is shown with the circle in

Figure 3. The elevations of points on that circle can be calculated by interpolating the elevations of nearby existing points in the LiDAR dataset. The number of directions water can flow to, represented in angle, is specified as an input variable for this algorithm. That is to say, the number of directions that water can flow to equal 360 degrees over the angle parameter. In Figure 3, for the simplicity of illustration, we have the angle differentiated by 45 degrees, which implies 8 directions that water can flow to. After interpolating the elevations of these 8 points (*(a_1,b_1)* to *(a_8,b_8)*) based on the LiDAR dataset, we can identify the one point among the eight, say *(a_4,b_4)*, having the lowest elevation. After comparing with the elevation at point *(a, b)*, if the elevation at *(a_4,b_4)* is higher than the elevation at point *(a, b)*, then, water should not flow anywhere since all the surround points have higher elevations than the initial point and point *(a, b)* should be a pit. However, if the elevation at *(a_4,b_4)* is lower than the elevation at point *(a, b)*, then, water should flow from point *(a, b)* to point *(a_4,b_4)*. In this way, our algorithm captures the water flow process and can derive the flow direction from a given point to generate a vector representing the flow direction from the point. After balancing the trade-off between accuracy and available computational resources, the angle parameter is set as 15 degrees in this study, meaning water could flow to 24 different directions at any point.

4.3 Chain of vectors

In order to derive the drainage network, the continuation of the flow directions needs to be determined. Figure 4 shows the chain of vectors created for this purpose. At the end of the vector created as described in section 4.2, we further construct another vector representing the

continuation of the flow using the same idea described in section 4.2. Before the vector (water flow) reaches the pit point or boundary, we apply the same method described in section 4.2 and find the continuation of the flow recursively to construct a chain of vectors representing the flow direction starting from one initial point. In this case, starting from point *(a, b)*, the water flows sequentially to *(c, d)*, *(e, f),* and *(g, h)* as the end. This chain of vectors is recorded to represent the flow direction starting from point *(a, b)*. To extract drainage features from LiDAR data for the entire study area, we apply this process to create a chain of vectors for each integer pair *(x, y)* that lies within the area (Fig. 4). Inspired by the A* search algorithm (Ehlschlaeger, 1989; Metz et al. 2010), our method creates a vector chain to represent water flow from each grid point. Although the process of creating a vector to represent water flow direction appears to be similar, our method does not need to go to another point for each step of the vector creation process, making the method efficient and flexible for representing flow directions.

4.4 Tracking vector chains

By tracking the chains of vectors, we can identify the drainage areas where water flow rarely passes through the landscape. First, many 1m*1m bounding boxes in the coordinate system are created to fully cover the landscape. For each chain of vectors, we track down all bounding boxes it passes through. After tracking all the chains of vectors, each bounding box is assigned a value indicating the number of times it has been passed through by chains of vectors. The larger the number, the more times a bounding box would be passed through, indicting more water

flowing through the 1m*1m grid. After integrating the results from all the grids, a drainage network can be constructed.

4.5 Threshold

Thresholding is the final step, which separates those major stream channels from minor water flows. The drainage network we have generated through the process described in section 4.4 contains both major streamlines and minor flows. In order to separate those major stream channels from minor flows, we do thresholding using a high percentile for all of the grids to construct the drainage network and extract major streamlines.

5. Computational Intensity Analysis and Parallelization

Our method is computationally intensive. Therefore CyberGIS-Jupyter (Yin et al., 2018) deployed on a geospatial supercomputer called Virtual ROGER (Resourcing Open Geospatial Education and Research) (Wang, 2017) is exploited. The implementation of this method uses the Python programming language and CyberGIS-Jupyter. The core algorithm for the method is shown in the pseudocode in Figure 5. (Fig. 5)

The vector-based and point-level calculations of this algorithm lead to high computational intensity. In the following section, we will first analyse the time complexity of the algorithm. Then, a parallelization strategy, as well as the impact of the parallelization, will be analyzed.

5.1. Time Complexity

In order to assess the time complexity of the algorithm for a study area of *m*n* grid cells, several key parts need to be analysed: 1) create a data structure to hold the LiDAR data points, 2) find the elevation for any data point *(a, b)* based on the LiDAR point cloud, 3) create a vector chain for every data point, and 4) integrate all vector chains for post-processing.

5.1.1 Data Structure

Finding the elevation of any given point *(x, y)* based on the LiDAR point cloud is one of the key components in the algorithm. Combining with the non-raster-based characteristics of LiDAR data point, a straightforward method by directly obtaining the elevation of any point *(x, y)* based on the interpolation of three closest points was adopted. Instead of interpolating the original data point (LiDAR point cloud in this research) into a raster-based DEM and then doing another interpolation to capture the elevation of a data point as done by conventional methods, our method directly interpolates at the level of LiDAR point to avoid doing the interpolation twice for improving analysis accuracy.

To minimize the time complexity of computing the elevation of a given point *(a, b)*, a data structure of hash table is created. A central goal for using this hash table data structure is to resolve the computational complexity for searching, which is used in the bilinear interpolation

for finding the data points that lie within each 1*1 grid. When calculating the elevation at a specific point during the process of creating a vector chain, a search function is called to find the existing LiDAR points that lie close to the target point that needs to be interpolated. Thus, we are using the hash table data structure to store the LiDAR points to reduce the computational complexity for searching. For an *m*n* dataset, we create a hash table with its hash function set to *floor(lx)*n+floor(ly)*. In this case, as shown in Figure 6, for any point *(x, y)* in the LiDAR dataset, where *x* is the longitude measured in meters, *y* is the latitude measured in meters, this LiDAR point is stored in the hash table bucket where the key is *floor(lx)*n+floor(ly)*. In this case, for all the LiDAR data points in the dataset, we can map them to a hash table data structure where the time complexity for searching is *O(1)* (Fig. 6).

5.1.2 Find the elevation for any data point

In order to compute the elevation for any point *(a, b)*, the data points in the surrounding nine blocks are captured. Within these 9 blocks, once there are more than 3 points, we use the Euclidean distance to find the 3 closing points near point *(a, b)*. Otherwise, if there are less than 3 points in these 9 blocks, we extend the search range to the surrounding 25 (5*5) blocks, (7*7) blocks, and so on until we successfully find three LiDAR points near the initial point. However, due to the fact that there are on average 43 LiDAR points within one square meter, in most cases, there is at least one LiDAR point corresponding to one hash key value. The situation where there are fewer than 3 points in the surrounding 9 blocks rarely happens. With the 3 closest LiDAR points near point *(a, b)*, we use bilinear interpolation to calculate the elevation of the data point

at *(a, b)*. Consequently, the time complexity for computing the elevation for any data point is *O(1)*.

5.1.3 Create a vector chain for each data point

In order to assess the time complexity of creating a vector chain, first, we create a circle around the initial point *(a, b),* and then we find the elevation of the points in a circle around the initial point. Since the time complexity of calculating the elevation given any point is *O(1)*, and the number of points is fixed based on the angle parameter (see section 4.2), the process of creating a vector is *O(1)* as well. Second, the time complexity of computing the continuation of vectors is *O(1)*. In most cases, a vector chain does not reach the boundary of the study area before it gets stopped when hitting a pit. Consequently, the time complexity for creating a vector chain for each cell is *O(1)*. Last, for a study area with the length of *m* pixels and width of *n* pixels, combing with the previous analysis showing that creating a vector chain for each point is *O(1)* time complexity, the time complexity for creating a vector chain for all data points in the study area is *O(m*n)*.

*5.1.4 Integrating the vector chains on the m*n pixel map and post-processing*

All the vector chains need to be integrated to generate the final drainage network. As is shown in section 4.4, we track all the vector chains for each cell. Since the number of vectors in each chain does not correspond to the size of the study area, the time complexity of integrating the vector

chains is *O(m*n)*. At the same time, the time complexity of post-processing is analogous to applying a kernel to the drainage system network, and thus is also *O(m*n)*. However, if we are investigating a large study area, the computational intensity can be challenging as each chain of vectors requires the calculation of the elevation of multiple data points.

In aggregate, the time complexity of the entire algorithm is *O(m*n)*. The time complexity of our algorithm is at the same level as the conventional methods of D8 and D-infinity, even though the algorithm has a higher computational intensity due to the free flow characteristics and the large volume of LiDAR data. Therefore, we study how to take advantage of high-performance parallel computing to resolve the computational intensity of the algorithm.

5.2. Parallelization

As assessed in section 5.1, the process of creating a chain of vectors for each point is the most time-consuming part of the algorithm. Figure 7 illustrates an example of using parallel computing to speed up this part. Suppose we have a study area of *m*n* pixels. First, we can divide the entire area into 4 sub-areas of size *(m/2)*(n/2)* pixels. For each sub-area, directly applying the algorithm as it is shown in the previous section will yield the exact same result since, for all the points in the study area, the process of creating a vector chain for each point is independent of each other as long as the pre-defined boundary is the same. In this way, we can divide the original computation job into 4 sub-jobs with each job running much faster than the original job. Each sub-job does computation for *(m/2)*(n/2)* pixels to derive *m*n/4* chains of

vectors. For each chain of vectors generated, we treat every vector equally and keep track of the pixels that vectors pass through in the entire study area of *m*n* pixels to get the final output. In theory, the running time of 4 sub-jobs in total equals the running time of applying the algorithm onto the entire study area. Using parallel computing, in this case, will speed up the computing process of creating a chain of vectors. As a result, given sufficient computing resources, we can speed up the algorithm by applying the techniques of parallel computing (Fig. 7).

5.3. Impact of Parallelization

Based on Amdahl's law, the speedup of the job depends on the proportion of the task that can be faster due to the adoption of parallel computing (p in the formula below) and the speedup of the part that can be used for parallel computing (s in the formula below). In this research, p represents the process of creating a chain of vectors in this algorithm which roughly equals the number of nodes available multiplied by the amount of processors we are using on each node. (Equation 1). For example, our algorithm achieved a speedup of 16.2 using 25 threads.

$$Speedup(s) = \frac{1}{(1 - p) + \frac{p}{s}}$$

6. Result

This section describes the validation of the model and assesses the impact of the angle parameter. First, the outputs of the model, which are the chains of the vectors, will be integrated to generate a map of the study area regarding those locations where water flows to and other locations where

water unlikely to reach. Also, based on the original map derived from the integration, we threshold the amount of the chains of vectors flowing through each grid. Then, we extract the main streamline from the result of the model and compare the result with the reference data provided by USGS. Section 6.2 discusses the impact of the angle parameter on the final result. As we have suggested in section 4.2, the smaller the angle parameter is, the more directions water can flow, which in theory makes the model more accurate. Section 6.2 also addresses the extent of model improvement and assesses the trade-off between model accuracy and running time.

6.1 Validation

After integrating the chains of vectors for each grid and tracking the amount of chains of vectors flowing through each grid, our method generates intermediate results for the drainage system network depicted by the top left side of Figure 8. These results capture the number of times each pixel has been passed through by vector chains. For a given pixel, the more times the pixel is passed through by vector chains, the more likely the pixel represents a part of a pit or drainage network. For the purpose of illustrative visualization, the result after thresholding is shown (Fig. 8). As we can see on the top left figure, the grids from the left middle of the study area to the right middle are lighter than the other grids, indicating that those areas are places where abundant water flow through. Consequently, we may assume from the drainage system network that the main streamline lies in the middle of the study area from the left middle area to the right middle area. We have also noticed that many of those lighter areas with abundant water flowing through are perpendicular to our imagery main streamline, showing that the water from two sides where

the elevation is higher is flowing to the main streamline. Generally speaking, the closer a grid to the main streamline in the middle, the lighter the grid becomes, indicating more water flowing through.

Apart from deriving the original drainage system network, we have also done thresholding to make the map more straightforward and easy to understand. Specifically, the thresholding is applied over the amount of water chains flowing through each grid. For example, if the threshold is 20, then only those grids with more than 20 chains of vectors having passed through are shown in white color while the other grids where the amount of chains do not reach the threshold remain black (Figure 8). As the threshold is reduced, there are more grids colored in white as the water distribution and drainage system network become more visible in the study area.

To validate our results, we compare the drainage network from our vector-based algorithm with the drainage network generated using D8 and D∞ with fine-resolution (1-ft) LiDAR-based DEM, as shown in Figure 8g and Figure 8h. The computation of D8 and D∞ is conducted via LAS Dataset To Raster function and FlowDirection function in ArcGIS. Compared with the drainage network generated with our vector-based algorithm, the results generated using D8 and D∞ with high-resolution LiDAR-based DEM lack granularity as a majority of the area is regarded as streamlines. As a consequence, it is difficult to extract major streamlines especially for places where streamlines are wide based on high-resolution data. Moreover, compared with our vector-based method, the drainage networks generated from existing raster-based methods

are more drifted from the reference data, indicating a larger bias in the corresponding algorithmic processes (Fig. 8).

After generating the drainage system network and applying the threshold, we aim to identify the main streamline based on the original drainage system network. According to Figure 8, it is difficult to find the main streamline since there are some bumps in the river bed in real-world scenarios causing discontinuity of the water flow as shown in Figure 9. These bump locations in the river bed are colored in black; but in the streamline detection, that might be regarded as part of the main streamline. As a result, in order to find the main streamline from the drainage system network, we apply a Kernel Density Estimation (KDE) method by firstly regarding each endpoint in the chain of vectors we got from the previous step as an event. Then, we do the data cleaning by filtering some noisy points. Figure 9 compares the reference data with the result from KDE and the drainage system network with the left part representing the reference data of the real river network in the study area; the middle part represents the bump locations derived using our method; and the right part combines the reference data and the origin graph in Figure 8. It is worth noting that the top left corner and the bottom right corner in the middle part reflect noisy data and that the final KDE result is similar to the reference data (Fig. 9).

Figure 10 shows the streamline derived from the drainage system network and the reference data. In order to extract the streamline from the drainage system network, we first find the point in the study area where the most amount of water flowing through as the starting point. Then we find all the nearby grids around it and determine the grid with the most amount of water flowing

through among all the grids that are close to the initial point. After running this process iteratively, we can find a list of grids being highlighted as the representation of the main streamline. Figure 9 shows the result of streamline detection and reference data. As we can see, the result is similar to the reference data (Fig. 10).

Figure 11 illustrates the difference between the reference data and the result from our method. In order to measure the accuracy of the result, we use the distance between the reference data and the detected main streamline as the measurement. For each grid point in the result data, we measure the distance between the reference data and that exact point. The result is shown in the figure, the mean value of the distance between the result data point and reference data is 1.64 meters, the variance is 1.53 while the maximum value of the distance is 5.39 meters. This indicates that the result of the main streamline is as far as 5.39 meters from the reference data and the average distance between the reference data and the result is 1.64 meters. Given the fact that the main streamline in the reference data only covers the center of the main streamline and there is potentially a water channel with a width larger than 1 meter, our result is desirable and the main streamline detection using the proposed algorithm is accurate in this study area (Fig. 11).

6.2 Impact of the angle parameter

As illustrated in Section 4.2, given different values of the angle parameter, the accuracy of the model varies due to the number of directions the water can flow to. Generally speaking, given a

smaller angle, the model will be more accurate because for each vector, there will be more directions available. However, the trade-off is that more computing resources are needed in order to support the input of smaller angles.

To assess this trade-off, we use a dataset with 30 degrees compared against 15 degrees for 25 *100*100* pixels subarea. The mean value of the difference using two different degrees ranges from 0 to 4 meters and the average mean value is around 2 meters while the largest variance reaches 50. As a result, the angle parameter does have a noticeable impact on the performance of the method. The angle parameter selection requires a balance between the availability of computing resources and desirable accuracy.

7. Conclusion

This research has developed a new method for analyzing drainage networks based on LiDAR data. Enabled by cyberGIS and high-performance computing, this method is designed to accurately delineate drainage networks using LiDAR point cloud. There are three major advantages of this new method compared to conventional methods. First, the method allows water to freely flow in any direction instead of limiting the flows to 4 or 8 directions as done by conventional methods. Second, this model is constructed based on LiDAR point cloud instead of DEMs. Lastly, the method can be parallelized to take advantage of high-performance computing for resolving computational intensity. After comparing our result with the reference data

provided by USGS, we find that the resulting drainage system network is valid and that streamlines can be accurately detected.

Our method does not remove pit holes from the original dataset, which is different from what we did for the DEM dataset. Although there are algorithms developed to generate a pit-free Canopy Height Model (CHM) with LiDAR data (Marcu et al., 2017; Khosravipour et al., 2013), there lacks a well-recognized pit-removal model for ground surface LiDAR data, unlike the case of DEM data (Wang et al., 2019). Another reason for not treating pit holes in this research is that our method is powerful in terms of finding a drainage system network and finding the main streamline even without doing pit removal. Furthermore, our method can remove pits by utilizing a proper threshold. Locations, where abundant water flows to but not within the main streamline, are the areas where the elevation is lower than the areas around them. Above all, our method is capable of achieving accurate and valid results with pits removed directly.

Future work needs to concentrate on using other interpolation methods such as bicubic interpolation for measuring the elevation of every data point. Investigation on parameter selection, including the angle and the length of the circle radius specified in the algorithm, can be another direction for further improving the accuracy of the method. Additional efforts should also be devoted to increasing computational scalability for handling massive LiDAR data. We plan to evaluate our method in other study areas through extensive comparison with DEM-based algorithms to gain further understanding of the generalizability and performance of the method.

Acknowledgment

This research is supported in part by US Geological Survey under grant number G14AC00244 and the National Science Foundation under grant numbers: 1443080 and 1664119. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF or USGS. Our computational work used Virtual ROGER, which is a cyberGIS supercomputer supported by the CyberGIS center for Advanced Digital and Spatial Studies and the School of Earth, Society and Environment at the University of Illinois at Urbana-Champaign.

References

Anderson, D. L., & Ames, D. P. (2011). A Method for Extracting Stream Channel Flow Paths from LiDAR Point Cloud Data. Journal of Spatial Hydrology, Vol 11, No 1.

Ariza-Villaverde, A.B., Jiménez-Hornero, F.J., Gutiérrez de Ravé, E. (2015). Influence of DEM resolution on drainage network extraction: A multifractal analysis, Geomorphology, Volume 241, 2015, Pages 243-254, ISSN 0169-555X, https://doi.org/10.1016/j.geomorph.2015.03.040.

Burger, G., Sitzenfrei, R., Kleidorfer, M., & Rauch, W. (2014). Parallel flow routing in SWMM 5. Environmental Modelling & Software, 53, 27-34.

Chen, Y., Zhou, Q., Li, S., Meng, F., Bi, X., Wilson J.P., Xing, Z., Qi, J., Li, Q., & Zhang, C. (2014). The simulation of surface flow dynamics using a flow-path network model, International Journal of Geographical Information Science, 28:11, 2242-2260.

Comer, P., Faber-Langendoen, D., Evans, R., Gawler, S., Josse, C., Kittel, G., Menard, S., Pyne, M., Reid, M., Schulz, K., Snow, K., Teague, J. (2003). Ecological Systems of the United States, A Working Classification of U.S. Terrestrial Systems NatureServe, Arlington, Va (2003), p. 75

Costa-Cabral, M. and Burges, S.. (1994). Digital elevation model networks (DEMON): A model of flow over hillslopes for computation of contributing and dispersal area. Water Resour. Res., vol. 30(6), 1994, pp. 1681-1692.

Ehlschlaeger, C. (1989). Using the A* search algorithm to develop hydrologic models from digital elevation data, Proceedings of International Geographic Information Systems (IGIS) Symposium, Baltimore, MD, USA, 275–281, 1989.

Freeman, T.G.. (1991). Calculating catchment area with divergent flow based on a regular grid. Computers & Geosciences, vol. 17(3), 1991, pp. 413-422.

Goulden, T., Hopkinson, C., Jamieson, R., and Sterling, S. (2014), Sensitivity of watershed attributes to spatial resolution and interpolation method of LiDAR DEMs in three distinct landscapes, Water Resour. Res., 50, 1908– 1927, doi:10.1002/2013WR013846.

Hopkinson, C., Hayashi, M. and Peddle, D. (2009), Comparing alpine watershed attributes from LiDAR, Photogrammetric, and Contour-based Digital Elevation Models. Hydrol. Process., 23: 451-463. https://doi.org/10.1002/hyp.7155

Jasiewicz J., Metz M.. (2011). A new GRASS GIS toolkit for Hortonian analysis of drainage networks. Computers & Geosciences, Volume 37, Issue 8, 2011, Pages 1162-1173, ISSN 0098-3004,

Khosravipour A., Skidmore A. , Isenburg M., Wang T., & Hussin Y. (2013). Development of an algorithm to generate a LiDAR pit-free canopy height model. SilviLaser 2013, October 9-11, 2013 – Beijing, China. Paper Number: SL2013-030

Liu, J., Zhu, A. X., Liu, Y., Zhu, T., & Qin, C. Z. (2014). A layered approach to parallel computing for spatially distributed hydrological modeling. Environmental Modelling & Software, 51, 221-227.

Lu, F., Song, J., Cao, X., & Zhu, X. (2012). CPU/GPU computing for long-wave radiation physics on large GPU clusters. Computers & Geosciences, 41, 47-55.

Lukač, N., & Žalik, B. (2013). GPU-based roofs' solar potential estimation using LiDAR data. Computers & Geosciences, 52, 34-41.

Lyu, F., Yin, D., Padmanabhan, A., Choi Y., Goodall J., Castronova, A., Tarboton, D., & Wang, S. (2019) . Reproducible Hydrological Modeling with CyberGIS-Jupyter: A Case Study on SUMMA. In Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning) (PEARC'19). ACM Press, New York, NY, Article 21. DOI: https://doi.org/10.1145/3332186.3333052

Marcu, C., Stătescu, F. & Iurist, N. (2017). A GIS-Based Algorithm to Generate a Lidar Pit-Free Canopy Height Model. Present Environment and Sustainable Development, 11(2), pp. 89-95. Retrieved 23 Mar. 2020, from doi:10.1515/pesd-2017-0027

Metz, M., Mitasova, H., and Harmon, R.S.. (2010). Accurate stream extraction from large, radar-based elevation models.Hydrol. Earth Syst. Sci. Discuss., 7, 3213–3235,

Mower, J. E. (1994). Data-parallel procedures for drainage basin analysis. Computers & Geosciences, 20(9), 1365-1378.

Murphy, P.N.C., Ogilvie, J., Meng, F.-R. and Arp, P. (2008), Stream network modelling using lidar and photogrammetric digital elevation models: a comparison and field verification. Hydrol. Process., 22: 1747-1754. https://doi.org/10.1002/hyp.6770

Ortega, L., & Rueda, A. (2010). Parallel drainage network computation on CUDA. Computers & Geosciences, 36(2), 171-178.

Polidori, L., & Chorowicz, J. (1993). Comparison of bilinear and Brownian interpolation for digital elevation models. ISPRS Journal of Photogrammetry and Remote Sensing, 48(2), 18-23.

Qin, C. Z., & Zhan, L. (2012). Parallelizing flow-accumulation calculations on graphics processing units—From iterative DEM preprocessing algorithm to recursive multiple-flow-direction algorithm. Computers & Geosciences, 43, 7-16.

Rahil, S., Xu, Z., Sugumaran, R., and Oliveira, S.(2016). Parallel landscape driven data reduction spatial interpolation algorithm for big LiDAR data. ISPRS International Journal of Geo-Information, 5, 97.

Roelens, J., Höfle, B., Dondeyne, S., Orshoven, J., Diels, J. (2018). Drainage ditch extraction from airborne LiDAR point clouds, ISPRS Journal of Photogrammetry and Remote Sensing, Volume 146, 2018, Pages 409-420, ISSN 0924-2716, https://doi.org/10.1016/j.isprsjprs.2018.10.014.

Salach, A.; Bakuła, K.; Pilarska, M.; Ostrowski, W.; Górski, K., & Kurczyński, Z. (2018). Accuracy assessment of point clouds from LiDAR and dense image matching acquired using the UAV platform for DTM creation. ISPRS Int. J. Geo-Inf. 2018, 7, 342.

Steinbach, M., & Hemmerling, R. (2012). Accelerating batch processing of spatial raster analysis using GPU. Computers & Geosciences, 45, 212-220.

Sten, J., Lilja, H., Hyväluoma, J., Westerholm, J., & Aspnäs, M. (2016). Parallel flow accumulation algorithms for graphical processing units with application to RUSLE model. Computers & Geosciences, 89, 88-95.

Tarboton, D. G. (1997). A new method for the determination of flow directions and upslope areas in grid digital elevation models. Water Resources Research, 33(2), 309–319.

Tesfa, T. K., Tarboton, D. G., Watson, D. W., Schreuders, K. A., Baker, M. E., & Wallace, R. M. (2011). Extraction of hydrological proximity measures from DEMs using parallel processing. Environmental Modelling & Software, 26(12), 1696-1709.

Wang, S., and Armstrong, M. (2009). A theoretical approach to the use of cyberinfrastructure in geographical analysis. International Journal of Geographical Information Science, 23 (2), 169–193

Wang, H., Zhou, Y., Fu, X., Gao, J., & Wang, G. (2012). Maximum speedup ratio curve (MSC) in parallel computing of the binary-tree-based drainage network. Computers & Geosciences, 38(1), 127-135.

Wang, S. (2017). CyberGIS. In International Encyclopedia of Geography: People, the Earth, Environment and Technology. John Wiley & Sons, Ltd. http://dx.doi.org/10.1002/9781118786352.wbieg0931

Wang, L., & Ai, T. (2018). The Comparison of drainage network extraction between square and hexagonal grid-based DEM. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences, Volume XLII-4, 2018, pp.687-692

Wang, S., Zhong, Y., & Wang, E. (2019). An integrated GIS platform architecture for spatiotemporal big data. Future Generation Computer Systems, 94, 160-172.

Wang, Y. J., Qin, C. Z., & Zhu, A. X. (2019). Review on algorithms of dealing with depressions in grid DEM. Annals of GIS, 25(2), 83-97.

Xu, Z., Guan, K., Casler, N., Peng, B., & Wang, S. (2018). A 3D convolutional neural network method for land cover classification using LiDAR and multi-temporal Landsat imagery. ISPRS Journal of Photogrammetry and Remote Sensing, 144, 423-434.

Xu, Z., Wang, S., Stanislawski, L., Jiang, Z., Jaroenchai, N., Sainju, A., Shavers, E., Usery, E., Chen, L., Li, Z., Su, B. (2021). An attention U-Net model for detection of fine-scale hydrologic streamlines, Environmental Modelling & Software, Volume 140, 2021, 104992, ISSN 1364-8152, https://doi.org/10.1016/j.envsoft.2021.104992.

Yin, D., Liu, Y., Hu, H., Terstriep, J., Hong, X., Padmanabhan, A., & Wang, S. (2018). CyberGIS-Jupyter for reproducible and scalable geospatial analytics. Concurrency and Computation: Practice & Experience, Volume 31, Issue 11.

Zhang, F., & Zhou, Q. (2019) Parallelization of the flow-path network model using a particle-set strategy. International Journal of Geographical Information Science, 33(10), 1984-2010.

Zhang, Y. J., Jha, M., Gu, R., Wensheng, L., & Alin, L. (2012). A DEM-based parallel computing hydrodynamic and transport model. River Research and Applications, 28(5), 647-658.

Figures