Journal of Applied Remote Sensing

RemoteSensing.SPIEDigitalLibrary.org

Graph-based approach for airborne light detection and ranging segmentation

David L. Vilariño
José C. Cabaleiro
Jorge Martínez
Francisco F. Rivera
Tomás F. Pena

SPIE.

David L. Vilariño, José C. Cabaleiro, Jorge Martínez, Francisco F. Rivera, Tomás F. Pena, "Graph-based approach for airborne light detection and ranging segmentation," *J. Appl. Remote Sens.* **11**(1), 015020 (2017), doi: 10.1117/1.JRS.11.015020.

Graph-based approach for airborne light detection and ranging segmentation

David L. Vilariño,* José C. Cabaleiro, Jorge Martínez, Francisco F. Rivera, and Tomás F. Pena
University of Santiago de Compostela, Centro Singular de Investigación en Tecnoloxías da Información, Santiago de Compostela, Spain

Abstract. A graph-based segmentation technique has been tailored to segment airborne LiDAR points which, unlike images, are irregularly distributed. In our method, every LiDAR point is labeled as a node and interconnected as a graph extended to its neighborhood, defined in a 4-D feature space: the spatial coordinates (x, y, z) and the reflection intensity. The interconnections between pairs of neighboring nodes are weighted based on the distance in the feature space. The segmentation consists of an iterative process of classification of nodes into homogeneous groups based on their similarity. This approach is intended to be part of a complete system for the classification of structures from LiDAR point clouds in applications needing fast response times. In this sense, a study of the performance/accuracy trade-off has been performed, extracting some conclusions about the benefits of the proposed solution. In addition, an interlaced graph-based approach is proposed to increase the reliability in general purpose segmentations. © *2017 Society of Photo-Optical Instrumentation Engineers (SPIE)* [DOI: 10.1117/1.JRS.11.015020]

Keywords: airborne LiDAR; point cloud segmentation; graph processing.

Paper 16844P received Nov. 10, 2016; accepted for publication Feb. 22, 2017; published online Mar. 10, 2017.

1 Introduction

Light detection and ranging (LiDAR) is one of the most popular technologies for airborne remote sensing. Like image processing domain, segmentation of LiDAR point clouds is an important stage in a processing chain for registration, classification, or reconstruction of different objects and structures. Fast and accurate segmentation methods are keys in an overall system. The aim is to group points with similar attributes, groups that can be identified as structures or part of structures in subsequent processing stages. This abstraction layer makes the object identification in high-level processing easier and significantly reduces the amount of data when compared to the original data set and, consequently, the computational cost of subsequent processing stages.

In the literature, different strategies to approach this task can be found. Most of them are adapted from methods formerly introduced for image segmentation. General-purpose methods are usually data-driven strategies. Some examples of data-driven image segmentation techniques successfully adapted to 3-D point cloud processing are clustering,[1] region growing,[2,3] or graph-based segmentation.[4,5] When the geometric characteristics of the objects under interest are known, model-based segmentation techniques are often preferred. Some examples are those approaches based on the Hough transform or the RANSAC fitting.[6–8]

Segmentation techniques perform in a feature space, instead of the image plane, fit to airborne laser scanned data because of their unstructured nature. In this work, a graph-based technique originally intended for image processing has been tailored for the segmentation of airborne LiDAR points. Particularly, the segmentation algorithm of Felzenswalb and Huttenlocher[9] was

*Address all correspondence to: David L. Vilariño, E-mail: david.vilarino@usc.es

1931-3195/2017/$25.00 © 2017 SPIE

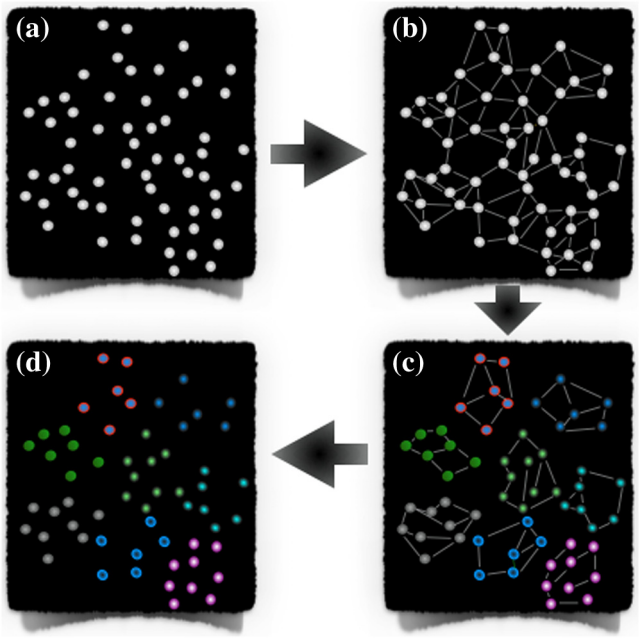

Fig. 1 Illustration of graph-based segmentation. (a) Lidar points are nodes of the graph. (b) The initial graph is created from the relation between neighboring nodes. (c) The graph is rearranged in disjoint groups which represent the final segmentation (d).

adapted. In this algorithm, every single LiDAR point is labeled as a node, and these nodes are interconnected as an undirected graph extended to the neighborhood defined in a 4-D feature space (x, y, z, and intensity). The edges that characterize the interconnections between two nodes are weighted based on their distance in this feature space. Therefore, pairs of nodes for which their LiDAR points present high similarity in the 4-D feature space are connected by an edge with low weight. Both neighborhood structure and distance between nodes are computed by using a k-d tree partitioning strategy. Figure 1 shows the idea behind the graph-based segmentation. Nodes are initially connected to their neighboring nodes. Then, these edges are cutoff or kept in their places based on the relation between the groups containing the nodes associated to the edges. The result is a set of disjoint groups representing the final segmentation.

This paper is based on the ideas introduced in Ref. 10 including a variation which we name "interlaced graph-based segmentation." This new technique, suitable for multicore processing, improves the robustness and accuracy when the size of the structures of interest is not a constraint, or the characteristics of the dataset are not known *a priori*.

The rest of the paper is organized as follows: In Sec. 2, a detailed description of the segmentation algorithm is carried out. Then, in Sec. 3, an evaluation of the algorithm's performance is presented. In Sec. 4, the interlaced scheme suitable for multicore processing is analyzed. Finally, the conclusions are given in Sec. 5.

2 Graph-Based Segmentation

The segmentation algorithm starts from a graph composed of nodes and edges. The nodes correspond to LiDAR points in the original 3-D point cloud. The edges are links between nodes according with the neighborhood model (see Sec. 2.1), and the weights are determined from the features of the points.

Figure 2 shows the block diagram of the algorithm. The input of the algorithm is the LiDAR points file, and the output is the segmentation. The algorithm consists of three main steps, which are described in the following sections. A number of parameters are also used: the neighbor count, the weight of each of the features under consideration (x, y, z, and intensity), and a calibration parameter K.

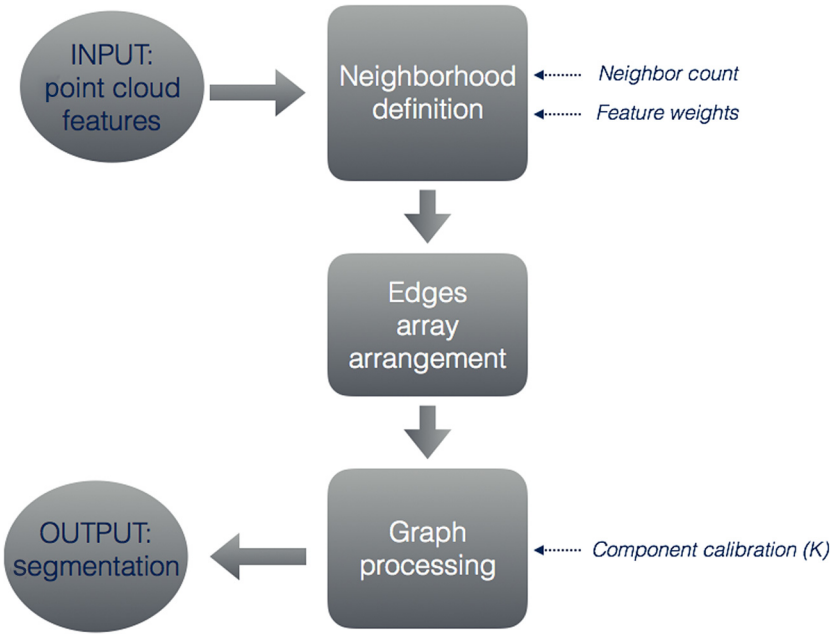

Fig. 2 Block diagram of the segmentation algorithm. The small labels indicate user-configurable parameters.

2.1 *Neighborhood Model*

For the graph construction, it is necessary to define a neighborhood model and the weights assigned to the links between nodes. Unlike the image domain, LiDAR data sets present irregular distributions, which make the neighborhood representation critical for subsequent data processing.[11] Different neighborhood models can be used in the LiDAR data processing. Among the most common approaches are the Delaunay triangulation[12] and the K-nearest neighbors (KNN).[4,13] Other popular approaches define neighbors as points within a bounding box (usually a cylinder or a sphere). These methods are very sensitive to the parameter selection (number of neighbors, radii, and orientation of bounding box, etc.), which greatly depends on the data characteristics, and, in particular, on the point density. The accuracy of some tasks, such as the computation of normal vectors or some segmentation techniques like region growing, are highly dependent on the neighborhood definition. However, for the graph-based segmentation, the set of connections produces indirect links between nodes providing robustness with suboptimal neighborhood definitions.

In this work, we have used a KNN search based on the k-d tree partitioning,[14] a very efficient nearest neighbor algorithm in low dimensionality spaces. Particularly, we have used the implementation included in the open source library FLANN fast library for approximate nearest neighbors optimized for fast processing.[15] This library is broadly used in the computer vision community as it provides different techniques to determine the neighborhood in feature spaces. In our experiments, we have considered the squared Euclidean distance in a feature space defined by the 3-D spatial coordinates (x, y, and z) and the intensity. Therefore, the links between nodes and their weights can be defined by the relative importance of any of these features. For example, high weights for x- and y-coordinates will lead to compact connected groups in the final segmentation, whereas high weights for intensity might lead to sparse distributed groups in which the key feature is the type of terrain.

2.2 *Graph Definition*

The segmentation consists of an iterative process of classification of nodes into homogeneous groups (also called components) based on their similarity in the feature space. Initially, we define as many components as nodes in the graph. Then, the edges connecting these LiDAR points are

arranged and evaluated in ascending order. For a given edge, the minimum spanning tree (MST) of the components associated with the nodes connected by the edge under analysis is obtained. If the weight of the edge is smaller than the lowest weight in the previously calculated MSTs, the components are merged (strong edge). Otherwise, the edge is broken (weak edge). This decision can be formalized as

$$\text{edge}(C_i, C_j) = \begin{cases} \text{strong} & \text{if } w_{\text{edge}} < \min[\text{Int}(C_i), \text{Int}(C_j)] \\ \text{weak} & \text{otherwise} \end{cases}, \tag{1}$$

where $\text{edge}(C_i, C_j)$ is the edge between component i and j, w_{edge} is the weight associated with that edge, and $\text{Int}(C_i)$ represents the internal weight of component C_i defined as the largest edge of the MST connecting the nodes of C_i. Note that the internal weight of the initial components is zero as they contain only one node.

To encourage the merge between components in the first iterations, an offset is added to the internal weight of the components. This offset represents a powerful tool to configure the segmentation according to the application. It can be associated with the shape or the size of the components. In our experiments, this parameter is used to control the strength of the edges and the size of the final components (as explained in Sec. 3). The proposed default of this offset is

$$\text{offset}(C_i) = K/\text{size}(C_i), \tag{2}$$

where K is the component calibration parameter shown in Fig. 2, and $\text{size}(C_i)$ represents the number of nodes assigned to component C_i. An important issue is that the MST of any new component should be recalculated to obtain the new internal weight (the highest weight in the MST), which would produce a high computational cost. Nevertheless, this is not necessary, as the internal weight can be updated by just comparing the internal weights of the components that have been merged to produce the new one and the weight of the considered edge. The highest of these three values will correspond to the internal weight of the new component.

3 Algorithm Evaluation

In this section, results of our proposal for the segmentation of LiDAR point clouds are discussed. As a case study, we have used a point cloud subset of 686,103 points with a density of 9 points/m^2 (Fig. 3). This set provides a high variability of landscape relief as well as different regions and structures, including small and regular constructions, extensive forest, large roads, and others.

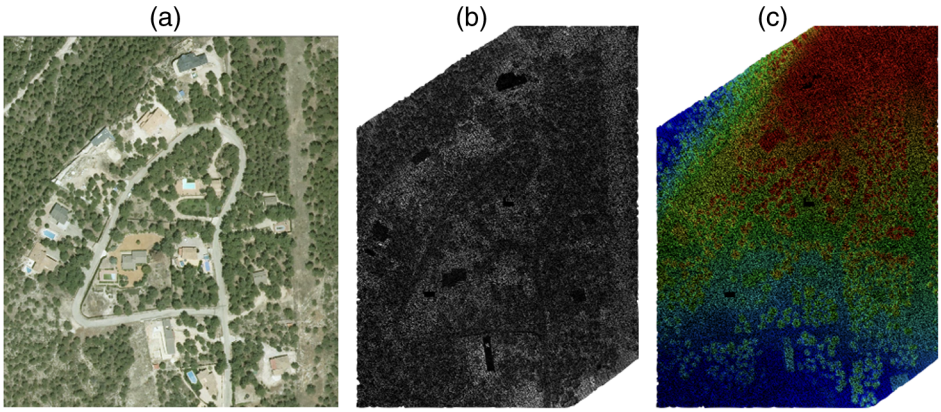

Fig. 3 Data set for validation: (a) orthophoto, (b) intensity map, (c) elevation map (from blue to red).

In order to perform the segmentation, it is needed to configure the input parameters (see Fig. 2): weights for x, y, z, and intensity, neighbors' count, and component calibration. For the experiments, we have used $w_x = 0.5$, $w_y = 0.5$, $w_z = 1$ and $w_I = 0.1$. Concerning the component calibration, we have considered the following equation:

$$\text{offset}(C_i) = \begin{cases} K/\text{size}(C_i) & \text{if } \text{size}(C_i) < \xi \\ K/[10 \times \text{size}(C_i)] & \text{if } \text{size}(C_i) > \xi \end{cases}, \tag{3}$$

where $\xi = 20{,}000$, $K = 100$ for edges under the average weight, and $K = 50$ for edges over the average weight. This strategy facilitates the merge of small components in the initial steps and prevents undersegmentation. ξ represents a powerful tool if the sizes of structures of interest are known *a priori*. For example, if we are interested in small buildings ξ can be calibrated accordingly. However, this parameter might be a limitation for general purpose segmentation since it is obtained experimentally and depends on the dataset. To deal with this issue, a variation of the graph-based segmentation which does not require this parameter is proposed in Sec. 4.

In order to illustrate the capabilities of this segmentation technique, Fig. 4 shows the result of the segmentation considering 50 neighbors (connecting every point in the initial graph). Note that the road with a "4" shape is correctly segmented in only one group and, at the same time, small buildings are well segmented, too. Furthermore, it can also be observed that extensive tree areas are segmented in large groups with some parts disconnected each other in the spatial domain.

Some applications require fast segmentation even at the expense of accuracy. However, loss of accuracy might be hard to manage in subsequent classification steps. In these situations, it is necessary to achieve a trade-off between computation speed and reliability. The main parameter related with both speed and accuracy is the neighbor count to construct the graph. The higher the number of neighbors, the higher the number of edges to be processed, which leads to slow

Fig. 4 Example of automatic segmentation using the graph-based approach.

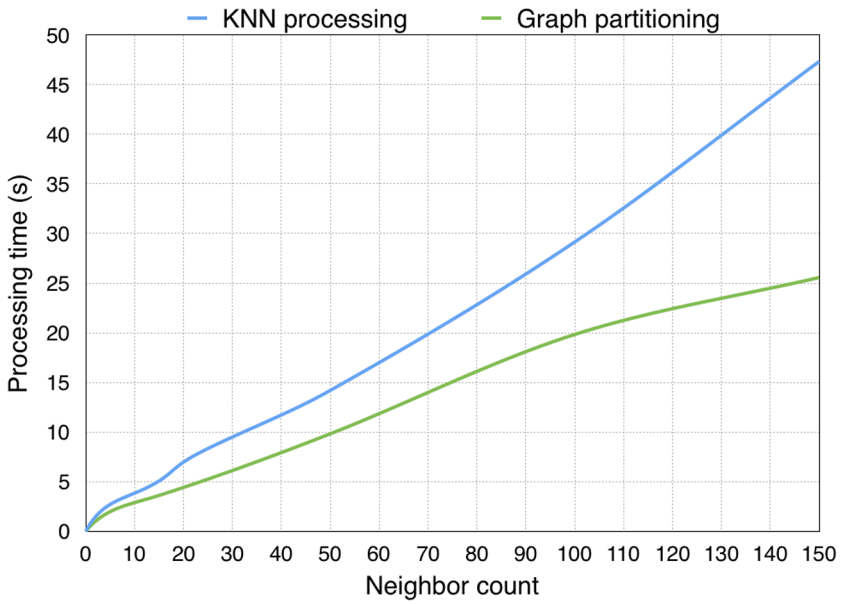

Fig. 5 Computation time of KNN estimation and graph partitioning.

computations in the graph organization. Figure 5 shows the influence of the neighbor count on the computation speed. The impact of the neighbor count is more noticeable in the KNN computation. The experiments have been carried out on an Intel Core i7-2600 at 3.4 GHz.

In Fig. 6, the relation between the resulting segmentation and the number of neighbors is illustrated. In this case, only groups with more than 15 points have been taken into account. Note that the number of groups quickly stabilizes with a neighbor count over five.

Table 1 includes numerical data for segmentation results with different neighbor counts. Percentage of segmented points refers to points included in groups with more than 15 points. For this dataset, smaller groups are considered irrelevant.

In our experiments, neighbor counts between 10 and 20 provide the best trade-off between computation cost and accuracy. Higher neighbor counts produce a rapid increase of the computation time with no significant improvement in the segmentation quality.

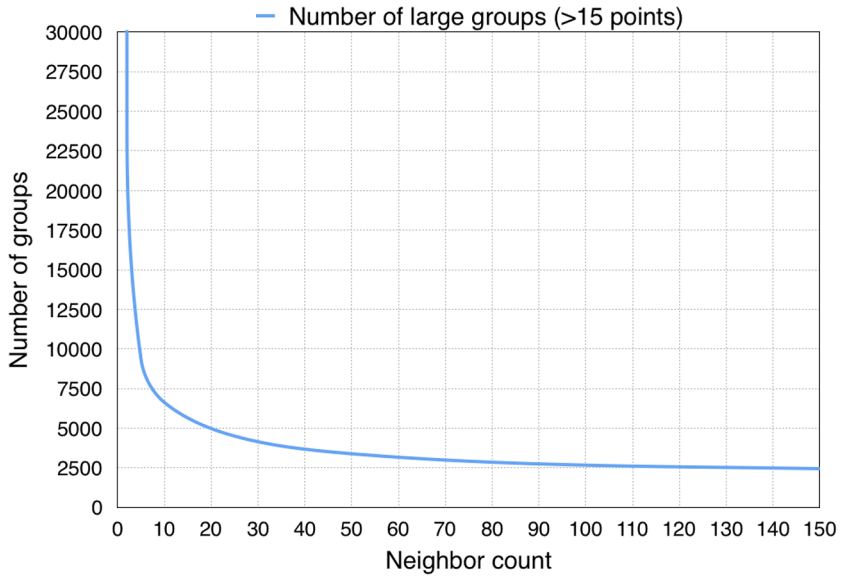

Fig. 6 Number of groups from data segmentation with different neighbor counts.

Table 1 Segmentation results versus computation time for different neighbor counts in a dataset with 686,103 points.

Neighbor count	2	5	10	25	50	100	150
KNN processing time (s)	1.88	2.71	4.26	8.83	14.19	29.15	47.34
Graph processing time (s)	1.23	2.12	2.84	5.32	9.82	19.83	25.57
Number of groups	23,667	9344	6621	4499	3385	2665	2444
Number of large groups (>15 points)	3954	1197	1016	1011	974	945	968
Percentage of segmented points (%)	83.93	96.67	97.68	98.45	98.80	99.00	99.10

4 Interlaced Graph

In practical situations, a valid segmentation should be as close as possible to the optimal solution. However, it is difficult, if not impossible, to identify the optimal segmentation. The segmentation quality is usually tested implicitly in subsequent classification steps where it is easier to measure the accuracy to classify objects of interest. For example, a road split in several segments can be easily reconstructed, or some parts of objects, like roofs or trees might be dealt in subsequent steps to recognize a building or a forest. In this sense, it is usually better to have an oversegmentation than an undersegmentation. Subsequent classification stages can deal with oversegmented areas, but it is rather more difficult to identify different objects within the same region from segmentation. The graph-based segmentation appears to be efficient to manage nonuniformly distributed point clouds. However, the final graph arrangement tends to undersegmentation. If two nodes (LiDAR points) are joined in the same group, all the nodes in the original groups of these nodes are also included in that group. The granularity of the segmentation can be controlled with the ξ parameter [see Eq. (3)]. This parameter constrains the size of the groups in terms of number of nodes. However, ξ might have to be experimentally determined or tuned relying on the previous knowledge of the input data.

With the purpose of providing robustness to the segmentation, we propose an interlaced scheme for the graph arrangement. The procedure of this new scheme is depicted in Fig. 7, and it is as follows: after the edges are ordered in increased order of weight, P graphs are constructed using all the nodes, but only with the P'th part of edges for each graph. These graphs are then arranged following the same procedure as with the single-graph approach.

As a result, P partial segmentations are available after the processing. The regions from all those segmentations are then combined to produce the final segmentation. This combination is performed by a hash on the group labels of the LiDAR points (one label per segmentation).

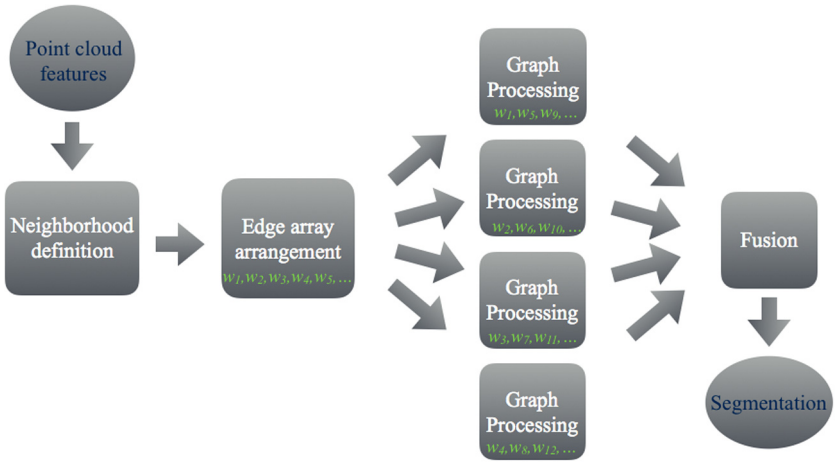

Fig. 7 Block diagram of the interlaced graph approach. The w_i are the ordered edge weights.

Algorithm 1 Interlaced graph (assuming N nodes, M edges, and P threads).

Data: N Lidar points

Result: P partial segmentations (S_i) and a final segmentation (S_F)

begin

  Search neighbors and assign weights for edges;

  Sort edges from low to high weights $(w_1, w_2, \ldots, w_M)$;

  for $i \leftarrow 1$ to P do

    Init graph G_i with N nodes and M/P edges $(w_i, w_{i+P}, w_{i+2P}, \ldots, w_{M-(P-i)})$;

    for $w_{\text{edge}} \leftarrow w_i$ to $w_{M-(P-i)}$ do

      if $w_{\text{edge}} =$ strong (see Eq.1) then

        Merge components of nodes connected with the edge;

        Calculate the internal weight of the new component;

      end

    end

  end

  Final segmentation $\leftarrow S_F = \bigcap_{i=1}^{P} S_i$;

end

This way, points with coincidence in all labels are put together in the same group in the final segmentation. A schematic description is shown in Algorithm 1.

The higher the number of partial segmentations, the higher the granularity of the final segmentation will be. In our experiments, we have found that $P = 4$ represents a good trade-off between accuracy and performance. Figures 8 and 9 show an example of processing a LiDAR point cloud with this interlaced graph-based approach. The point cloud has a million points with a average density of 9 points/m^2. Figure 8 shows the partial segmentations assuming a partitioning of the edge array in four parts in a round-robin fashion. Figure 9 shows the result of the combination of the four partial segmentations. Finally, Fig. 10 shows the result of the segmentation by using the single graph. In both interlaced and single-graph cases, Eq. (2) instead of Eq. (3) has been considered. Therefore, no additional constraint is imposed to the groups' size. For the other parameters, we have considered $w_x = 0.5$, $w_y = 0.5$, $w_z = 1$ and $w_I = 0.1$, as in the experiments shown in Sec. 3 and a neighboring count of 50.

Figures 11 and 12 show the results of the segmentation of a LiDAR point cloud corresponding to an urban area different from the previous dataset. Particularly, it corresponds to the German city of Vaihingen.[16] The point cloud contains 2.5 million points with a density of 4 points/m^2. In this segmentation, we have used the same parameters as in the previous rural area. Without external control of the group size [Eq. (3)], the single-graph approach tends to produce undersegmentation. However, the interlaced approach provides a valid segmentation under the same conditions.

Concerning execution times, the algorithm of the interlaced graph approach has been coded in such a way that the external "for" loop in Algorithm 1 can be fully executed in parallel. Segmentation results in Figs. 9 and 10 require 43 s on an Intel Core i7-2600 at 3.4 GHz using OpenMP[17] with four processing threads. In the case of the dataset in Figs. 11 and 12, the computation time is 88 s for the single-graph segmentation and 73 s for the interlaced approach. The computing overhead associated with the fusion step represents <0.01% of the overall execution time. The computing time strongly depends on the number of edges that are not broken in the graph as they produce the merger of groups and the subsequent computation

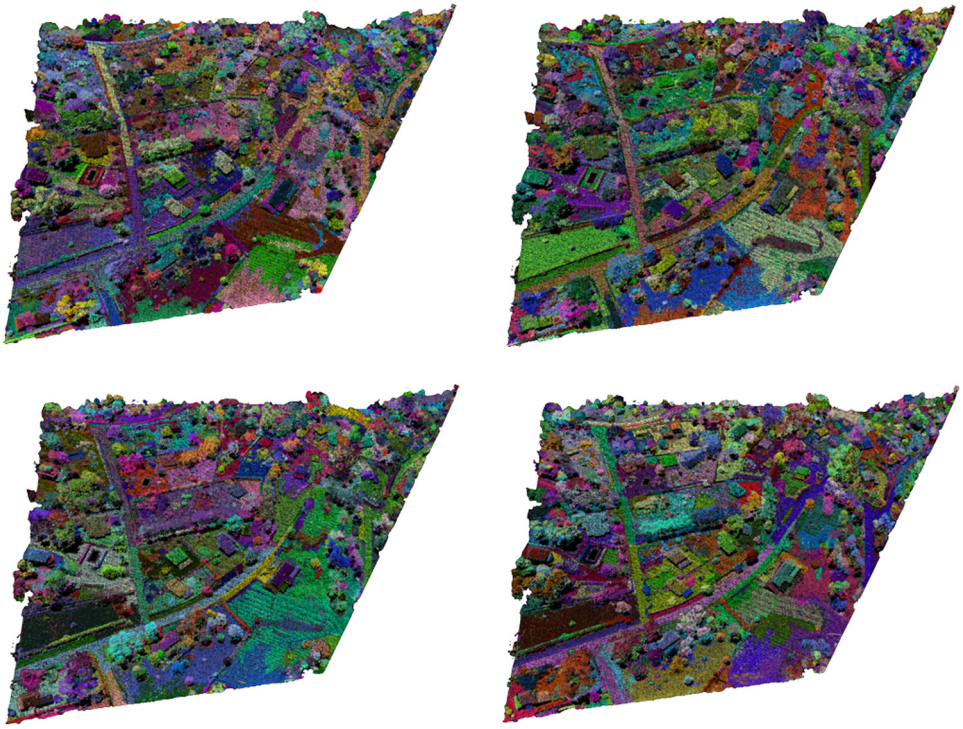

Fig. 8 Partial segmentations considering four graphs in the interlaced approach.

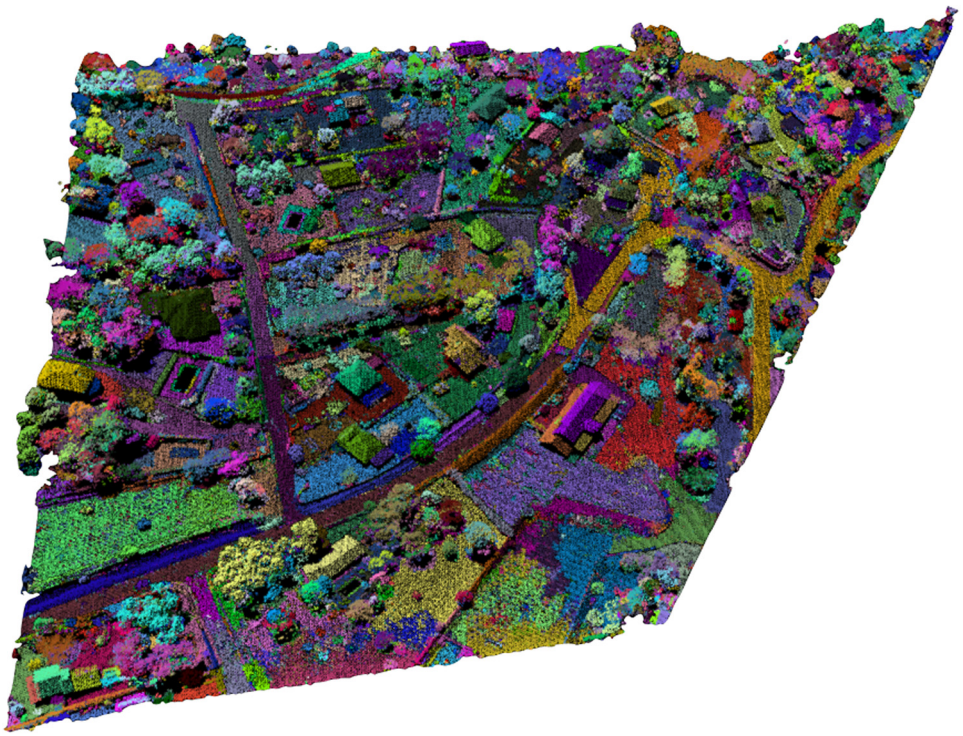

Fig. 9 Example of automatic segmentation using the interlaced graph-based approach (5108 groups).

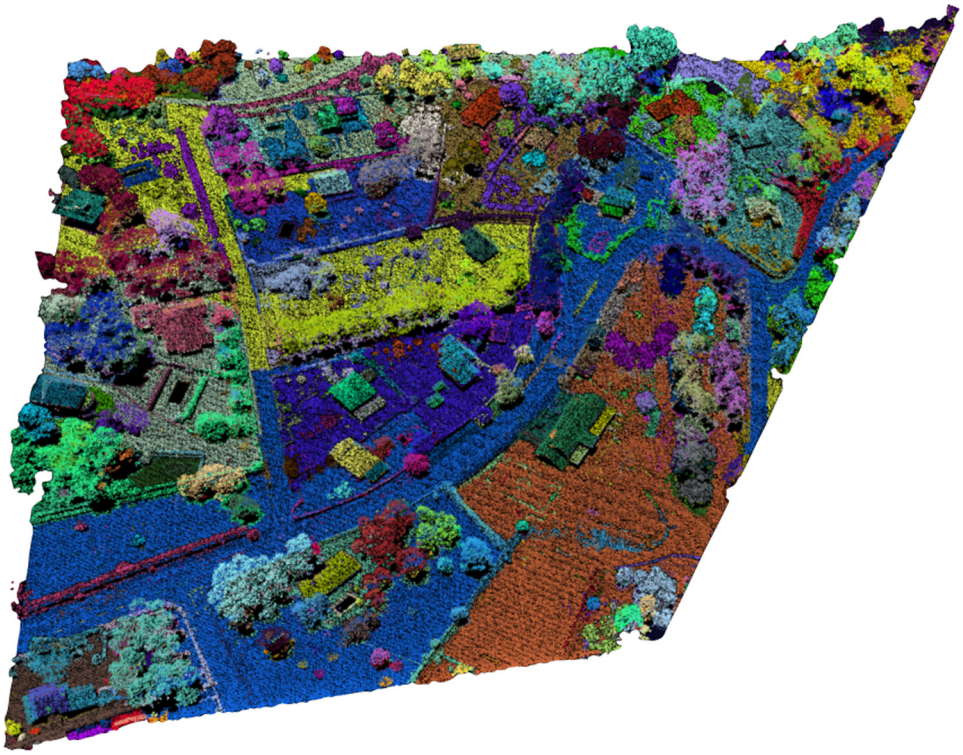

Fig. 10 Example of automatic segmentation using the single-graph-based approach (851 groups).

of the internal weight of the new groups. The probability of keeping an edge is inversely proportional to its position in the ordered array of weights. Therefore, the interlaced scheme for the parallel approach represents the most balanced workload distribution among threads. Table 2 shows the number of edges that produce a merger between groups (components) in the partial segmentation graphs of the examples mentioned in this section. Furthermore, the edge count for the single-graph approach is included. The higher degree of groups mergers in the single-graph approach penalizes the computing time with respect to the interlaced graph technique.

In order to illustrate the robustness of the segmentation algorithm, Fig. 13 shows the segmentation of a point cloud with only 0.5 points/m^2 density. The proposed segmentation technique produces good segmentations even with very low density of points. The dataset, provided by the Spanish Geography Institute through the National Plan of Aerial Ortophotography, corresponds to the City of Santiago de Compostela.

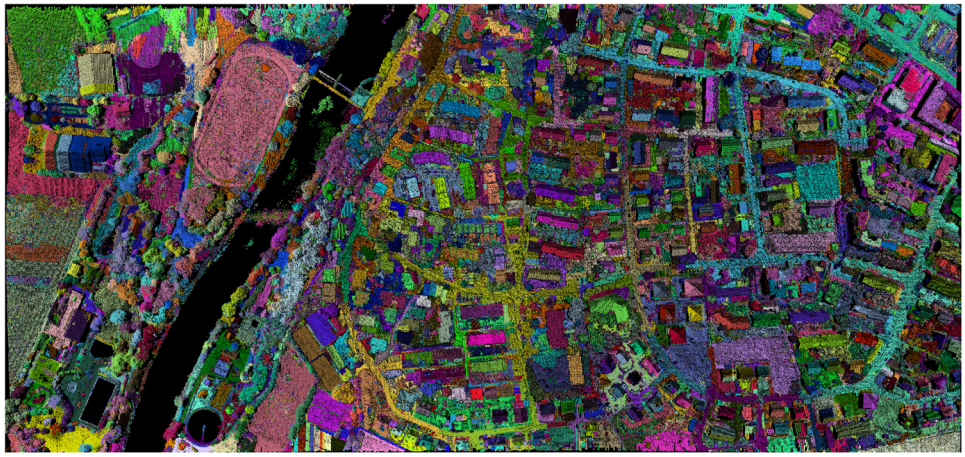

Fig. 11 Example of automatic segmentation using the interlaced graph-based approach (10,500 groups).

Fig. 12 Example of automatic segmentation using the single-graph-based approach (1645 groups).

Table 2 Number of edges that produce the merge of groups for the partial segmentations of the interlaced graph (inter 1, inter 2, inter 3, and inter 4) and the single graph (single) for the segmentations in Figs. 9 and 10 (dataset A) and Figs. 11 and 12 (dataset B).

Segmentation	Inter 1	Inter 2	Inter 3	Inter 4	Single
Dataset A	986,572	986,492	986,484	986,400	996,637
Dataset B	3,636,396	3,637,068	3,635,591	3,636,565	3,749,662

Fig. 13 Example of automatic segmentation on a dataset with very low-point density (0.5 points/m^2) using the interlaced-graph-based scheme.

5 Conclusions

In this paper, we describe a graph-based technique for the segmentation of airborne LiDAR point clouds. This approach can be situated in the middle between data and model-driven strategies, as it is possible to infer information about the structures to be segmented in the graph computation. The method provides fast and accurate results to be used in subsequent steps of classification and object recognition. However, one disadvantage of the graph methodology is the tendency toward undersegmentation which might be controlled by an external parameter that is dependent of the datasets. To avoid this, an interlaced segmentation technique is introduced. This strategy consists of the intersection of the groups from the partial segmentations, which reduces the probability of undersegmentation. Results of processing datasets with very different point densities show the robustness of the proposed segmentation technique.

Acknowledgments

This work has been partially funded by the Spanish Ministry of Science and Innovation with the Project No. TIN2016-76373-P and for the Xunta de Galicia with the Project Nos. GRC2014/008, R2016/037, and R2016/045. Moreover, the work has also been developed under the support of the European Network HIPEAC and the Spanish Network CAPAP-H. This work has also received financial support from the Consellería de Cultura, Educación e Ordenación Universitaria (accreditation 2016-2019, ED431G/08) and the European Regional Development Fund (ERDF). We also want to thank to the LaboraTe Research Group of the University of Santiago de Compostela.

References

1. A. Sampath and J. Shan, "Segmentation and reconstruction of polyhedral building roofs from aerial LiDAR point clouds," *IEEE Trans. Geosci. Remote Sens.* **48**(3), 1554–1567 (2010).
2. P. Dorninger and C. Nothegger, "3D segmentation of unstructured point clouds for building modelling," in *Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. **36**, pp. 191–196 (2007).
3. J. Martínez et al., "A rule-based classification from a region-growing segmentation of airborne LiDAR," *Proc. SPIE* **10004**, 100040F (2016).
4. A. Golovinskiy and T. Funkhouser, "Min-cut based segmentation of point clouds," in *IEEE Workshop on Search in 3D and Video (S3DV '09)*, pp. 39–46 (2009).
5. S. Ural and J. Shan, "Min-cut based segmentation of airborne LiDAR point clouds," in *Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. **39-B3**, pp. 167–172 (2012).
6. R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for point cloud shape detection," *Comput. Graphics Forum* **26**(2), 214–226 (2007).
7. G. Vosselman et al., "Recognizing structure in laser scanner point clouds," in *Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. **46**, No. 8/W2, pp. 33–38 (2004).
8. F. Tarsha-Kurdi, T. Landes, and P. Grussenmeyer, "Hough-transform and extended RANSAC algorithms for automatic detection of 3D building roof planes from LiDAR data," in *ISPRS Workshop on Laser Scanning and SilviLaser*, Vol. **36**, pp. 407–412 (2004).
9. P. Felzenswalb and D. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vision* **59**(2), 167–181 (2004).
10. D. Vilarino et al., "Graph-based segmentation of airborne LiDAR point clouds," *Proc. SPIE* **10004**, 100040I (2016).
11. S. Filin and N. Pfeifer, "Neighborhood systems for airborne laser data," *Photogramm. Eng. Remote Sens.* **71**(6), 743–755 (2005).
12. J. Zhang et al., "Contour clustering analysis for building reconstruction from LiDAR data," in *Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. **37**–**B3b**, pp. 355–360 (2008).

13. T. Rabbani, F. van den Heuvel, and G. Vosselman, "Segmentation of point clouds using smoothness constraint," in *Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. **36**, pp. 248–253 (2006).
14. J. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM* **18**(9), 509–517 (1975).
15. M. Muja and D. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Trans. Pattern Anal. Mach. Intell.* **36**(11), 2227–2240 (2014).
16. M. Cramer, "The DGPF-test on digital airborne camera evaluation—overview and test design," *Photogramm. Fernerkundung Geoinf.*, Vol. **2010**, No. 2, pp. 73–82 (2010).
17. L. Dagum and R. Enon, "OpenMP: an industry standard API for shared-memory programming," *IEEE Comput. Sci. Eng.* **5**(1), 46–55 (1998).

David L. Vilariño received his PhD from the University of Santiago de Compostela in 2001. Currently, he is part of the staff of the same university as an associate professor in the Department of Electronics and Computer Science and a senior researcher in the Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS). His research interest is the design and implementation of image and LiDAR data processing algorithms with fast and efficient computation as main concerns.

José C. Cabaleiro received his PhD from the University of Santiago de Compostela in 1994. Currently, he is an associate professor in the Department of Electronics and Computer Science at the University of Santiago de Compostela. His research interests include the architecture of parallel systems, the development of parallel algorithms for irregular problems and particularly for processing LiDAR data, and the prediction and improvement of the performance of parallel applications.

Jorge Martínez received his BS degree in computer engineering from the University of Santiago de Compostela in 2014 and his MS degree in high-performance computing from the University of A Coruña in 2016. He is a PhD student at the University of Santiago de Compostela. His research interests include image processing, LiDAR, and parallel computing.

Francisco F. Rivera is a full professor at the University of Santiago de Compostela, Spain. Throughout his career, he has supervised research and published extensively in the areas of computer-based applications, parallel processing, and computer architecture. His current research interests include compilation of irregular codes for parallel and distributed systems; the analysis and prediction of performance on parallel systems and the design of parallel algorithms; memory hierarchy optimizations, GIS.

Tomás F. Pena is an associate professor and a senior researcher in the Research Center at IT (CiTIUS), University of Santiago de Compostela, Spain. His main research lines include high-performance computing, architecture of parallel systems, development of parallel algorithms for clusters and supercomputers, prediction and improvement of the performance of parallel applications, development of applications and middleware for grid and cloud, and the use of big data technologies for scientific and NLP applications.