Journal of the Indian Society of Remote Sensing (September 2018) 46(9):1423–1431
https://doi.org/10.1007/s12524-018-0788-9

RESEARCH ARTICLE

A Minimum-Cost Path Model to the Bridge Extraction from Airborne LiDAR Point Clouds

Sheng Xu[1] · Shanshan Xu[1]

Received: 20 December 2017 / Accepted: 3 June 2018 / Published online: 22 June 2018

Abstract
Nowadays, bridges have played a significant role in human transportation networks. However, less attention has been paid to the bridge extraction from the countryside environment. This paper aims to propose a three-step method for the bridge extraction from airborne LiDAR point clouds. First, we propose a chain-code-based method to delimit land/water interface from the input scene. Second, we perform an angle testing process to extract candidate bridge points based on the shoreline delimitation result. Third, we calculate the cost of paths across the water body. A path whose cost is less than an adaptive threshold will be selected as a bridge path. The main contribution of this paper is that we formulate an energy function to calculate the cost of each potential bridge path. The optimal path, which achieves the minimum cost, is solved by the proposed minimum-cost path model. The developed extraction method does not rely on the geometric shape of rivers and works well in different types of bridges. Experiments show that the presented method succeeds to obtain all bridges in six small bridge scenes and one large complex scene, which are promising results in the bridge extraction.

Keywords Bridge extraction · ALS · Point clouds · Dynamic programming · Optimization

Introduction

A bridge is a structure built to span the gap between two lands without closing the way underneath, which is important to human transportation networks. The common key step in the inspection and health monitoring of bridges is the bridge extraction (Bian et al. 2012; Liu et al. 2010). Nowadays, the existing bridge extraction methods are mostly proposed for 2D images. Han et al. (2007) detect bridges from the satellite imagery using an integrated algorithm. Their integrated method is composed of two steps: the segmentation of the river and background based on a data driven-strategy and (2) the detection of bridges based on a knowledge-driven strategy. Chaudhuri and Samal (2008) achieve the bridge extraction from the multi-spectral imagery using a four-step approach. First, they classify pixels into water, concrete and background based on a supervised classification technique. Second, they extract potential bridge pixels by exploiting the geometric constraints of bridges. Third, they group those potential bridge pixels into candidate bridges based on their connectivity and geometric properties. Fourth, they filter false bridges based on the direction of the river. Gu et al. (2011) propose a hierarchy algorithm for extracting bridges from optical images. In the beginning, they set thresholds to obtain coarse water bodies using the edge information. Both the texture information and spatial coherence will be used to refine water bodies. After detecting water bodies, they regard objects across river regions as candidate bridges and remove false bridges based on the geometric information and the location between bridges and the river. The main problem in the bridge extraction from 2D images is that they are difficult to detect bridges flying over non-water regions, e.g. islands and ships.

Nowadays, ALS (airborne laser scanning) data have succeeded to collect the accurate 3D information of objects, which provides a new solution for achieving a better extraction result. However, until now, less attention has been paid to the bridge detection from LiDAR data.

In the work of Smeeckaert et al. (2013), they provide an automatic workflow for the classification of land and water.

✉ Shanshan Xu
shanshanxus@gmail.com

1 College of Information Science and Technology, Nanjing Forestry University, Nanjing, China

Features input for their supervised learning are based on 3D LiDAR point coordinates and flight-line information. Their results output by the classifier are merged with the contextual knowledge to refine the classification. They succeed to achieve a high classification accuracy in distinguishing the water and non-water region. However, they do not show how to extract individual bridges from the classification results. In the method of Sithole and Vosselman (2006), they detect bridges based on the topological information in the cross-section of a landscape. Their detection is adaptable to different bridge designs. However, their algorithm requires adding appropriate base points in the gaps left by the water body to identify seed points in the bridge detection. In the work of Duan et al. (2014), they propose a bridge extraction algorithm by the combination of an adaptive morphological filter and a skeleton extraction process. First, a morphological filter is designed to classify the data into ground points and non-ground points. Second, rivers are extracted based on the elevation features. Candidate bridges are located by using a morphological algorithm to thin the extracted skeleton. The final bridges are detected based on their shortest distance rule on the achieved skeleton lines. The problem is that the detection highly relies on the geometric information of bridges. Due to the mentioned drawbacks, the approaches described above may not provide the adequate extraction accuracy in the complex scene.

The target of this paper is to extract paths across water bodies based on the point coordinates only, i.e. no multi-echo, intensity, full-waveform or multi-spectral information, and provide a method for optimizing accurate bridges from airborne LiDAR point clouds in the countryside environment.

Methodology

The proposed bridge extraction is composed of three steps, including the delimitation of the water and land, the extraction of candidate bridge points and the optimization of bridge paths, which will be discussed in detail below.

Shoreline Delimitation

One challenge in the delimitation of the water and land is the removal of water body points. As mentioned in Mandlburger et al. (2015), not all the wavelength laser energy are absorbed by water. In the case of muddy and shallow water bodies, we may collect lots of returns. In the other cases, the return from the water region may be very minimal.

In the case of less return, we first cluster points based on the Euclidean distance, and then remove clusters containing fewer points, e.g. 200 points in our work. As shown in the first row of Fig. 1, points from the water body are removed effectively after the Euclidean clustering process. The following will provide a water point removal method based on the combination of the plane fitting and features filtering for the case of much return from water.

As we know, the water body is usually presented as a large horizontal plane in point clouds. Thus, plane regions can be regarded as candidate water bodies. The commonly used plane fitting approach in point cloud processing is the Random Sample Consensus (RANSAC), which can be implemented easily through the open source tool point cloud library (PCL)(Rusu and Cousins 2011). Extracted plane points may contain false water points. Therefore, we use the features mentioned in (Smeeckaert et al. 2013), which are calculated based on the elevation and density information, to remove false water bodies. In the countryside environment, the elevation of the water body is assumed to be low. Hence, if the elevation of a water point is larger than 2.5 m, we will remark it as a non-water point. Besides, the density of points from the water body region is relatively lower than other regions. Thus, if the density at a water point is larger than $0.3\,\mathrm{pt/m^2}$ in the $50\,\mathrm{m} \times 50\,\mathrm{m}$ region, we will remark it as a non-water point. As shown in the second row of Fig. 1, points from the water body are removed effectively based on the combination of the plane and features information. In the application, since we do not know the condition of returns from the water body, we will conduct all the above-mentioned water point removal steps.

Next, we will propose a method based on the chain-code approach (Freeman 1961) to delimit the shoreline, i.e. the contact between the land and water bodies. The shoreline is regarded as boundaries of lands, which will play a significant role in the constraint of candidate bridge points and provide the potential start and ending points of bridges.

Before we propose the shoreline delimitation method, we use the voxel-based representation technique for organizing non-water points. The volume of each voxel is $5\,\mathrm{m} \times 5\,\mathrm{m} \times 5\,\mathrm{m}$. Each voxel contains zero or more non-water points. The goal of the chain-code-based method is to separately encode each connected voxels in 3D point clouds. The idea is to select a voxel on the boundary and then move along the boundary of the region until returns to the start position. In our work, we prefer to choose the minimum number of directions in the 3D encoder as shown in Fig. 2a, including the left, right, up, down, front and back direction. For example, the 3D contour in Fig. 2b is encoded as $\{5, 1, 1, 2, 1, 2, 3, 4, 4, 5, 5, 4, 2, 6\}$. One can use more directions to refine the path, but this makes less improvement in the final bridge path extraction.

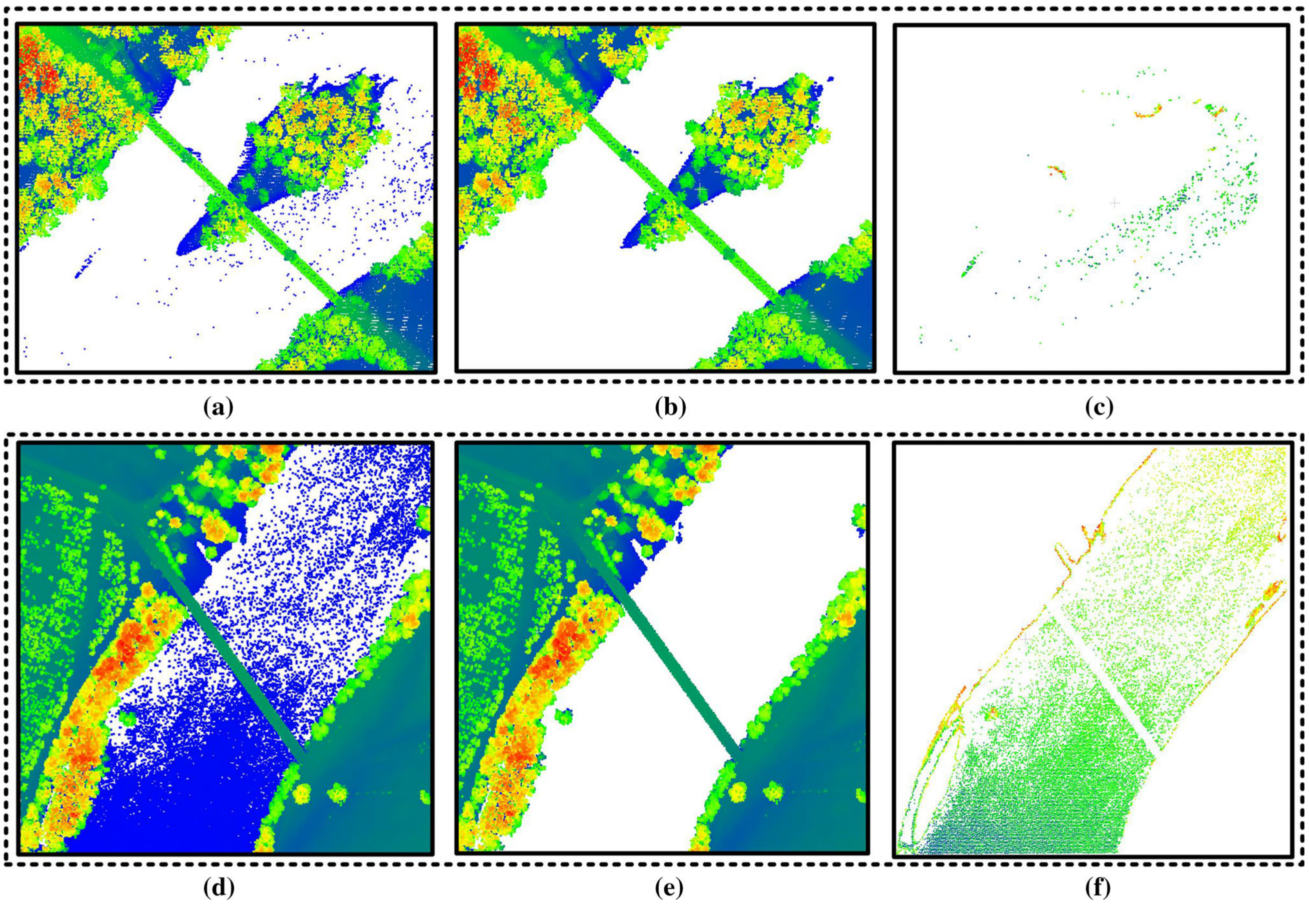
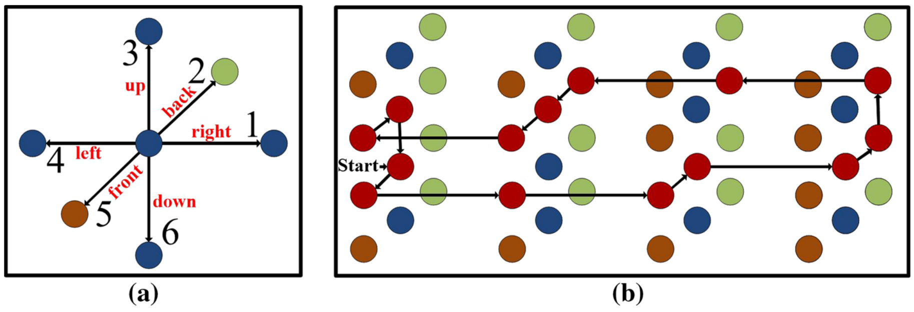


Fig. 1 The water point removal. **a** The input scene with less return from the water. **b** The removal result of **a**. **c** The removed water points from **a**. **d** The input scene with much return from the water. **e** The removal result of **d**. **f** The removed water points from **d**

Fig. 2 Encoder in the proposed chain-code-based method. **a** Selected six directions. **b** The formulation of a 3D contour using selected voxels (red) (color figure online)



Steps of the proposed chain-code-based method are as follows. The first step is the assignment of each voxel's value. If a voxel contains zero point, we set it as 0. If a voxel is located right above the water body region, we set it as 0. In all other cases, we set it as 1. The second step is the detection of boundary voxels. If two adjacent voxels have different values in the direction $\{1, 4, 2, 5\}$, i.e. the left, right, front and back direction, we mark them as boundary voxels. The third step is the extraction of the water body contour. We choose a start node among the boundary voxels and explore its neighbors in a fixed order to achieve the code for a closed contour. The fourth step is to repeat the above-mentioned chain-code-based method to extract all contours.

The example of the shoreline delimitation is shown in Fig. 3. Figure 3a shows the assignment of voxels' value after the water point removal. Figure 3b shows the extracted contours by the proposed chain-code-based method. It is worth noting that when the exploration of a contour arrives at the border of the input scene, it will continue to move along the border to formulate a closed contour.

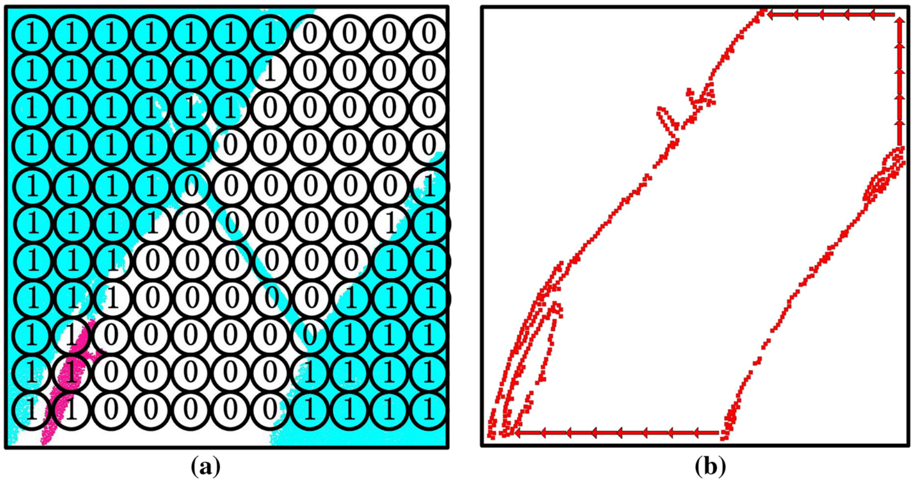


Fig. 3 Shoreline delimitation. **a** The assignment of voxels' value. **d** Extracted boundary contours

Candidate Bridge Points Detection

In our work, bridges are supposed to be across the water body region. Therefore, candidate bridge points are expected to fly over the surface of the water body. This section aims to propose a testing method for detecting candidate bridge points to reduce the computation complexity in the subsequent optimization.

First, we divide the contour into two disjoint shorelines as shown in Fig. 4a. Then, we test each non-water point to decide if it is a candidate bridge point or not. For a testing point x, denote its nearest point in the left and right shoreline as L_x and R_x, respectively. If x is not located above the surface of the water body, the angle $\angle L_x x R_x$ will be acute. If $\angle L_x x R_x$ is larger than $\pi/2$, x will be regarded as a candidate bridge point. Figure 4a shows the testing of four typical points, including two land points A and B, an island point C and a bridge point D. In the beginning of the testing, we find out the nearest left and right boundary points for each testing point, namely the point L_A and R_A for A, the point L_B and R_B for B, the point L_C and R_C for C, and the point L_D and R_D for D. $\angle L_A A R_A$ and $\angle L_B B R_B$ are less than $\pi/2$, which means that A and B are not interest points. $\angle L_C C R_C$ and $\angle L_D D R_D$ are larger than $\pi/2$, thus, both C and D belong to the candidate bridge points. An example of the extracted candidate bridge points is shown in Fig. 4b.

The proposed testing may fail in the case of thin and sharply curved rivers. In our experiment, the proposed strategy succeeds to detect all candidate bridge points when the width of the river is larger than 80 m.

Bridge Path Optimization

Paths across water bodies are regarded as candidate bridge paths. Each point from the left shoreline can be regarded as a potential start point of a bridge path as shown in Fig. 5a. To extract the optimal bridge path, we formulate the energy Eq. (1) to calculate the cost of each candidate bridge path.

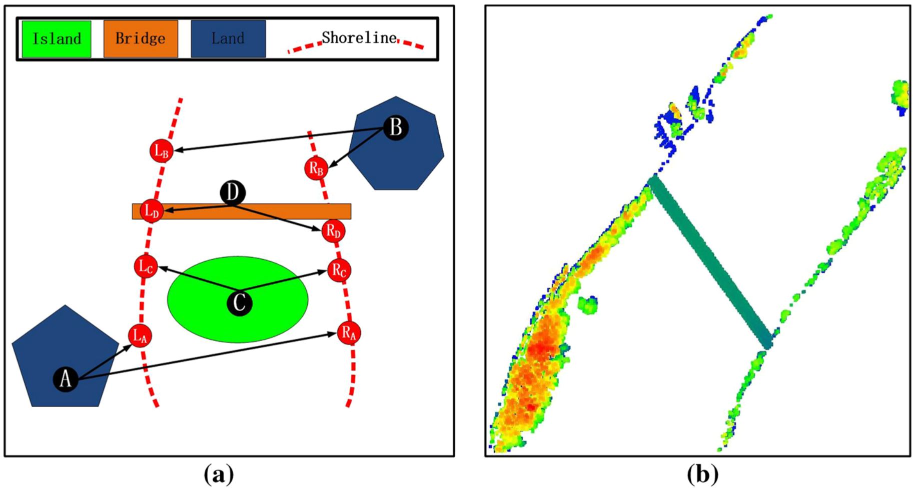


Fig. 4 Extraction of candidate bridge points. **a** The testing process. **b** Candidate bridge points

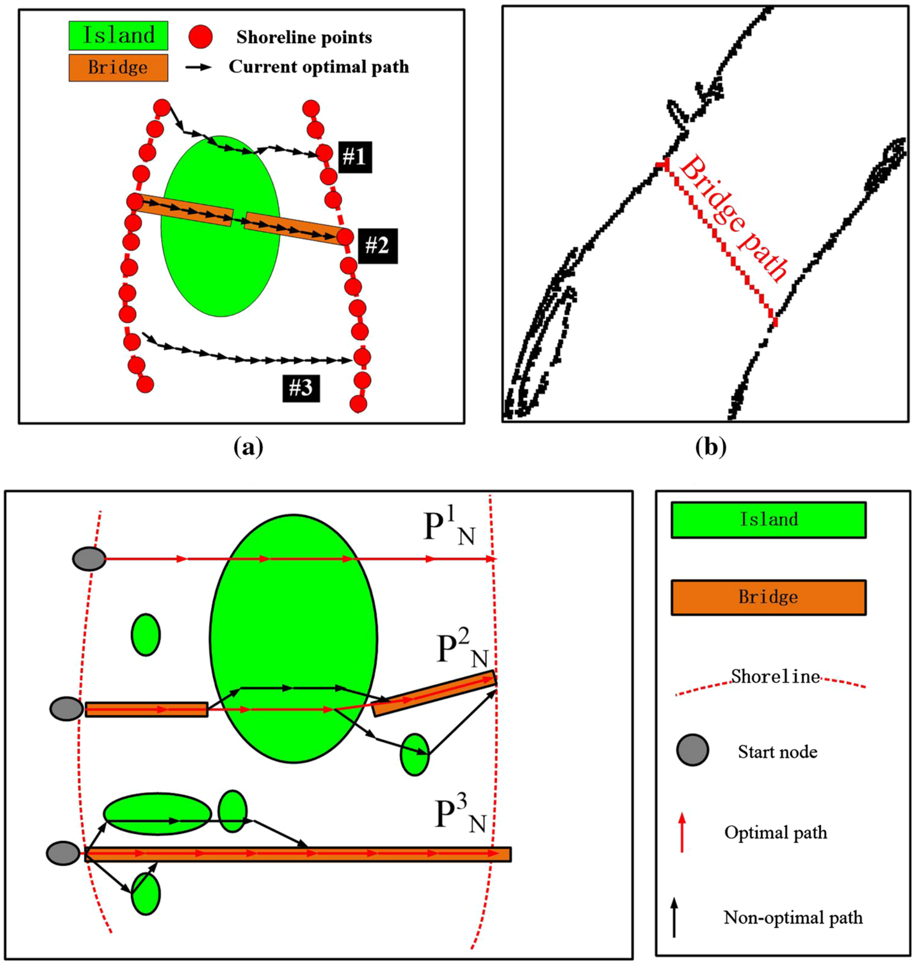


Fig. 5 Search for bridge paths. **a** The example of candidate bridge paths. **b** An optimal bridge path. **c** Illustration of multiple bridge paths

$$P_N = \sum_{i}^{N}(D(v_i) + \alpha S(v_i, v_{i-1})) \quad (1)$$

where N is the number of the selected points in the obtained candidate bridge path.

Assume that the main search direction of candidate bridge paths is from the left to the right of the input scene. Since bridges are very organized artifacts with high continuity and smoothness, we use the point v_{i-1} to represent the left nearest neighbor of v_i in the bridge path. v_1 is the selected start point for the current bridge path. The data term D is to calculate the cost of a bridge path containing all selected voxels. The direction term S aims to constrain the path direction and elevation. The coefficient α is used for balancing those two terms in the function. Terms are defined as follows.

$$D(v_i) = \begin{cases} 0, & dis(v_i, v_{i-1}) < d_0 \\ d, & \text{others} \end{cases} \quad (2)$$

$$S(v_i, v_{i-1}) = \sin < \mathbf{V}_{i-1,i}, \mathbf{V}_{i-2,i-1} > + |z_i - z_{i-1}|^2$$

In the data term calculation, $dis(v_i, v_{i-1})$ means the Euclidean distance between the point v_i and v_{i-1}. If the distance between two neighbors is smaller than the threshold d_0, the cost in data term is 0, otherwise, the path will be penalized by a user-defined cost d. In the direction term calculation, $\mathbf{V}_{i-1,i}$ means the vector pointing from the point v_{i-1} to v_i. The first term in the direction calculation is achieved by the sine of $\mathbf{V}_{i-1,i}$ and $\mathbf{V}_{i-2,i-1}$. z_i is the elevation of the point v_i and the second term in the direction calculation aims to make the bridge elevation consistent. In the calculation of D and S, if i is less than 2, we define $D(v_i)$ and $S(v_i, v_{i-1})$ as 0. Figure 5a shows the example of three candidate bridge paths. The optimal path is required to contain as many candidate bridge points as possible and keep a consistent direction and elevation. Therefore, the path #2 is the expected optimal path in Fig. 5a and will be regarded as the bridge path as shown in Fig. 5b. The following aims to propose a minimum-cost path model for finding the optimal bridge path.

In the beginning, we model a graph G for building the solution space as shown in Fig. 6a. Each node in G refers

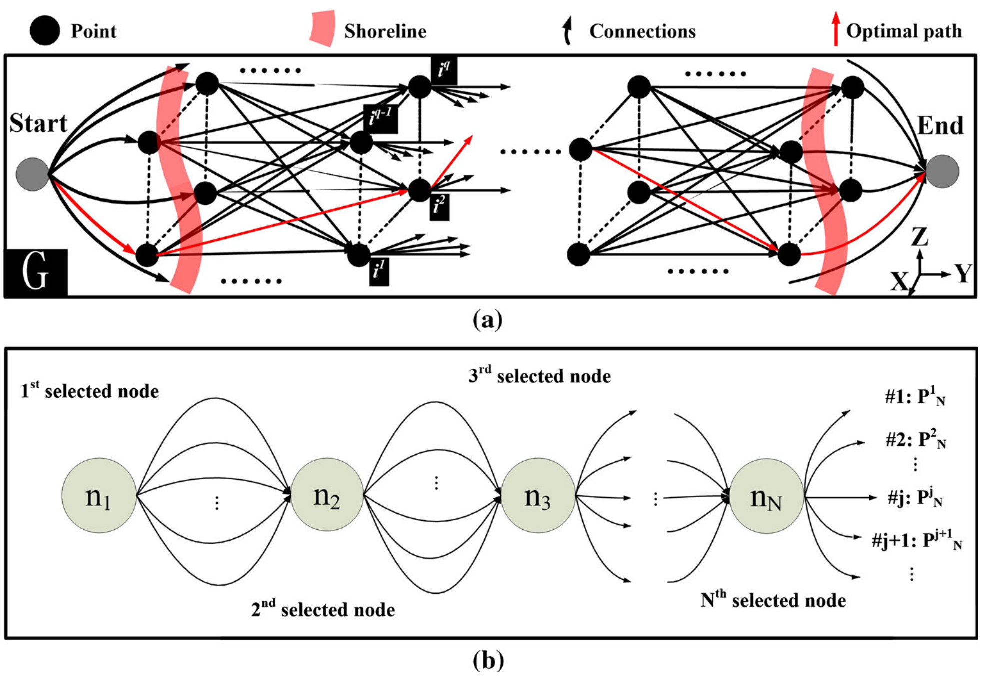


Fig. 6 Minimum-cost path model. **a** Search process. **b** Paths from the start node to the ending node

Table 1 The description of test datasets

#	Points ($\times 10^3$)	Size (m)	Density (pts/m^2)	Survey date (m/year)	Location (US)
(a)	2021	961 × 1453	1.45	11/2008	Vermont
(b)	657	235 × 333	8.40	10/2007	Maine
(c)	19	575 × 296	0.11	03/2011	Indianna
(d)	181	446 × 448	0.91	03/2011	Indianna
(e)	116	285 × 238	1.72	03/2011	Indianna
(f)	217	280 × 378	2.06	03/2011	Indianna
(g)	2516	1620 × 1624	0.96	03/2011	Indianna

to a point from the input candidate bridge points. The candidate bridge path is assumed to proceed from the left to the right of the input scene. Therefore, if the coordinate of the node n_{i+1} is (x, y, z), the coordinate of its prior node n_i should be (x^*, y^-, z^*), where $x^* \in [\min(X), \max(X)]$ and $z^* \in [\min(Z), \max(Z)]$. y^- is smaller than y and $dis(y, y^-) < d_0$. The graph G contains all possible paths from the start node to the ending node. Each path can represent a candidate bridge path.

The next step is to assign the penalty to the connection between nodes in G. The cost of each connection can be calculated by Eq. (2), and the sum cost of a path from the start node to the ending node is equal to P_N as mentioned in Eq. (1). From the start node n_1 to the ending node n_N, there will be exponential number of paths and each path $\#j$ in Fig. 6b corresponds a cost P^j_N.

The final step is to find the optimal path from all candidate paths to achieve the minimum cost P_N. Our optimization is based on the dynamic programming technique. Assume that we have found the optimal path from the start node to each node. When adding a new node n_{N+1} to G, the optimal path from the start node to n_{N+1} should contain the optimal path from the start node to its prior node n_N. Thus, to obtain the entire optimal path from the start node to n_{N+1}, we only need to find the minimum-cost connection from n_N to n_{N+1}. The corresponding P_{N+1} is calculated as

$$P_{N+1} = \min_{i=1}^{q}(P_{N^i} + D(v_{n+1}) + \alpha S(v_{n+1}, v_n)) \tag{3}$$

where q is the number of n_{N+1}'s prior nodes. Figure 6a shows the example of n_{i+1}'s prior nodes, including n_{i^1}, $n_{i^2}, \ldots,$ and n_{i^q}. Since P_{N^i} is known, the computation of P_{N+1} incurs a quite low complexity. Nodes in the minimum-cost path will be regarded as the selected points for obtaining the candidate bridge path.

Each path $\#j$ has a cost, which can be calculated by Eq. (1). Since an input scene may contain more than one bridge path, we choose each boundary point as a start node and conduct the proposed path search. Assume that a scene has Q candidate bridge paths and P^i_N is the cost of each

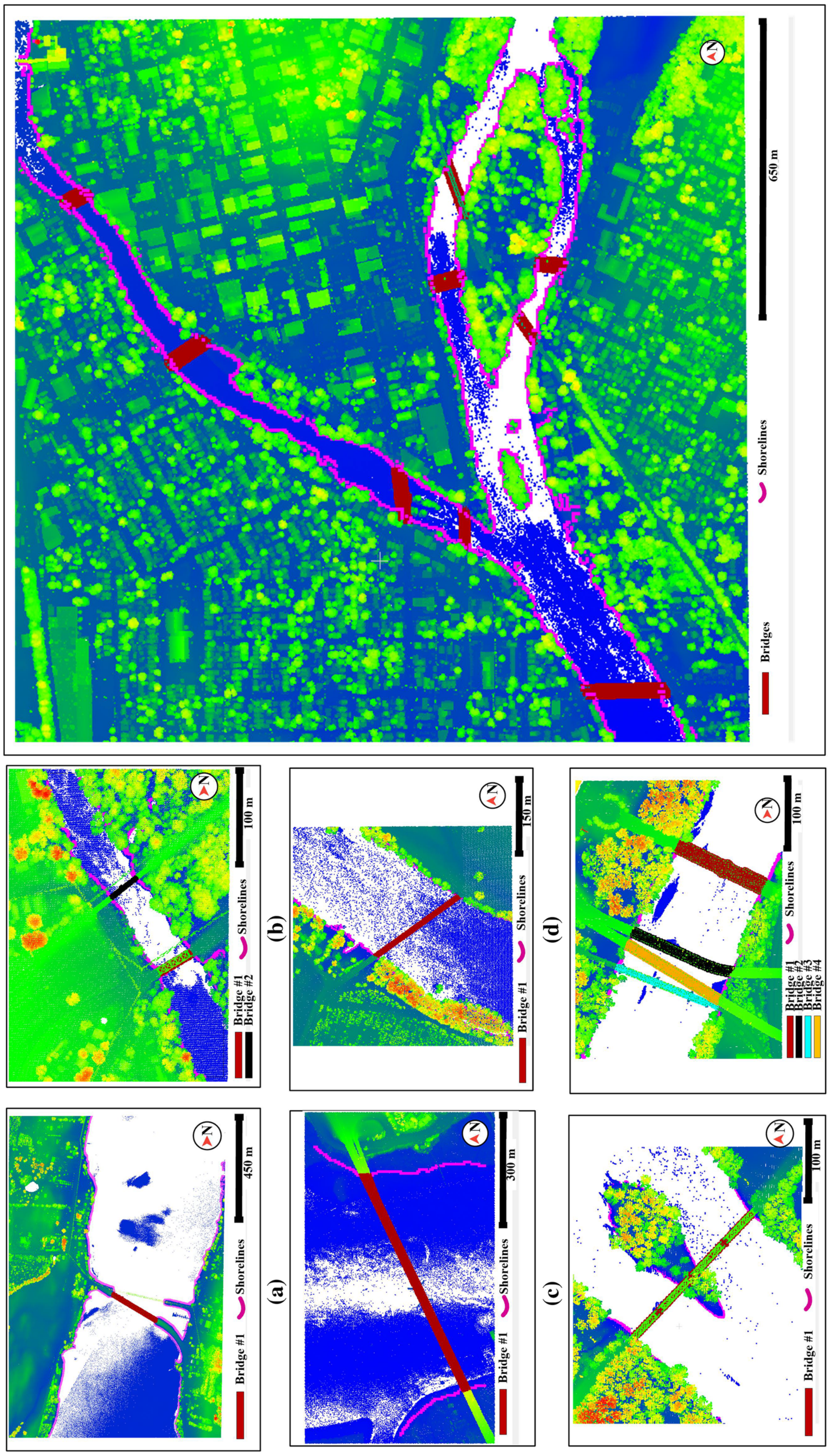


Fig. 7 The performance of the proposed method on different experimental scenes. **a** This scene contains one bridge. **b** This scene contains two bridges. **c** This scene contains one bridge. **d** This scene contains one bridge. **e** This scene contains one bridge. **f** This scene contains four bridges. **g** This large scene contains nine bridges

path, i.e. $i = 1, 2, \ldots, Q$. If the cost of a candidate path is less than the threshold p_0, i.e. $10 \times \min(P_N^i)$ in our work, this path will be stored as a bridge path. The illustration of searching multiple bridge paths is shown in Fig. 5c. Based on the above analysis, P_N^2 and P_N^3 are regarded as bridge paths.

Experiment and Evaluation

Because the number of bridges in a scene is usually low, bridges were manually selected from several scenes for the experiments described in this paper. To test the performance of our extraction, we applied it to different point cloud scenes. Table 1 shows the description of the experimental scenes, including the number of points, the size of the scene, the density of points, the survey date and location. The testing scenes are downloaded from OpenTopography (http://www.opentopography.org/). (a) is provided by US Geological Survey. (b) is provided by National Center for Airborne Laser Mapping. (c)–(g) are provided by IndianaMap Framework Data.

Our results of the bridge extraction are shown in Fig. 7. In Fig. 7a, there is a region extending to the water body, which is easy to be wrongly extracted as bridges by methods based on the cross section of the water region. In Fig. 7b, there are two bridges over a water body. The challenge is that points below bridges are very sparse but other water regions contain lots of points. In Fig. 7c, d, we show two common simple bridges in the countryside environment, including a railway bridge and cement bridge. The challenges in Fig. 7e is the involvement of an island below the bridge. The challenges in Fig. 7f include that four bridges are in different elevation and two of them are merged. In Fig. 7g, we show a large scene containing different types of bridges.

In the experiments, the average cost time across sites (a)–(f) is 57.03 s and the cost time in the site (g) is 283.82 s. Experiments were done on a Windows 10 Home 64-bit, Intel Core i5-7200U 2.5 GHz processor with 16 GB of RAM and computations were carried on Matlab R2018a. We succeed to accurately detect all bridges in Fig. 7, which indicates that the proposed method is very promising in the bridge extraction from ALS point clouds.

The test is performed on data with various densities. The point density in (c), (d) and (g) is less than 1.0pt/m^2, the density in (a), (e) and (f) is around 2.0pt/m^2, and the density in (b) is more than 8.0pt/m^2. In the case of the low density data, there will be more false water bodies, because the water point removal relies on the density information. In this case, we have to decrease the density threshold in the false water point removal process. The candidate bridge point extraction only depends on the shoreline delimitation, which is robust to the density. As we mentioned before, a bridge is very organized artifacts with high continuity and smoothness in ALS point clouds. The bridge optimization is also insensitive to the density, when users set appropriate parameters in the path cost calculation.

Table 2 Comparison of the accuracy of different methods

Method	Data	Total scenes	Total briges	B./S.	Acc. (%)
Sithole and Vosselman (2006)	ALS	6	8	1.33	100
Duan et al. (2014)	ALS	3	4	1.33	100
Proposed	ALS	7	19	2.71	100

The existing approaches to the bridge extraction did not provide detail detection accuracy. For the comparison, we evaluate extraction results based on the ratio of the correctly detect bridges. Evaluation results are shown in Table 2. In the comparison, all input data are ALS point clouds. The third column shows the number of experimental scenes in each method. The fourth column shows the number of bridges in each method. The fifth column shows the number of bridges in each scene and the last column shows the correctly detected bridge ratio.

All of the methods in Table 2 achieve 100% bridge extraction ratio. However, our test scene is more complex than others, which shows our robustness is better than others. We test the proposed algorithm on a large countryside scene and our B./S is much larger than other methods. In the method of Sithole and Vosselman (2006), bridges tend to be partially or completely removed in their filtering process when bridges are in different elevation. Figure 7f shows that our algorithm works well in dealing with the elevation difference. In the method of Duan et al. (2014), the extraction is invalid if there is no water point below the bridge. Figure 7e, f show that our algorithm does not rely on the completeness of water points.

The following is the analysis of parameters and the guideline for choosing parameters. As shown in Table 3, there are mainly 5 parameters in the bridge path optimization. k is to set the number of neighbor points in the k-nearest neighbors approach. It is used to find the prior nodes in the optimization. A larger k shows advantage in the case of incomplete bridges but incurs much more computation. α is the coefficient in the energy Eq. (1). It is used for balancing the data and smoothness term. A larger α is suitable for the classic bridges, i.e. bridges with very parallel edges, but the result may contain less candidate bridge points in the case of complex bridges. d_0 is the threshold to constrain the maximum distance between two neighbor nodes. It is used in the data term calculation.

Table 3 Parameters in the bridge path optimization

Parameters	Suggested	Unit
k	50	point
α	3.0	N/A
d_0	5.0	meter
d	1.0	N/A
p_0	$10\times \min(p_N^i)$	N/A

Similar to k, a larger d_0 brings much more prior nodes which may cause more computation in the optimization. d is the user-defined penalty. It is used in penalizing the selection of a non-neighbor node in the path. A larger d will increase the weight of data term and cause that the resulting path only connect as much candidate bridge points as possible in the optimization. p_0 is an adaptive threshold value. It is used in choosing multiple optimal bridge paths. A larger p_0 may reduce the miss extraction ratio but incur more false bridges. All suggested values in our experiments are shown in Table 3.

Conclusions

In this paper, we propose a three-step method for the bridge extraction from airborne LiDAR point clouds, including (1) the water point removal, and the land/water interface delimitation, (2) candidate bridge points extraction and (3) the bridge path optimization. Bridges are represented by paths across the water body and the cost of each path is measured by the proposed energy function. The optimization of the bridge path is achieved by the proposed minimum-cost path model using the dynamic programming technique. Our extraction requires airborne LiDAR point clouds only and the optimization is proceed based on the coordinate information without any human–computer interaction. Experiments show that the presented algorithm succeeds to extract all bridges in different scenes, which is competitive to other existing methods. Future work will show our performance in different environments, e.g. the urban and mountain areas, and focus on more complicated arch bridges.

Acknowledgements Authors would like to thank Prof.Ye and Dr.Zhu for helping the formulation and optimization of the energy function.

Funding National Key Research and Development Plan of China (2016YFD0600101), National Natural Science Foundation of China (31770591, 41701510), China Postdoctoral Science Foundation (2016M601823).

References

Bian, H., Bai, L., Chen, S.E., & Wang, S. G. (2012). Lidar based edge-detection for bridge defect identification. In *Proceedings of SPIE* (Vol. 8347, pp. 83,470X-1).

Chaudhuri, D., & Samal, A. (2008). An automatic bridge detection technique for multispectral images. *IEEE Transactions on Geoscience and Remote Sensing*, *46*(9), 2720–2727.

Duan, Y., Song, J., & Miao, Q. (2014). Bridge detection in light detecting and ranging data based on morphological filter and skeleton extraction. *Journal of Applied Remote Sensing*, *8*(1), 083,610-083,610.

Freeman, H. (1961). Techniques for the digital computer analysis of chain-encoded arbitrary plane curves. In *Proceedings of national electronics conference* (Vol. 17).

Gu, D. Y., Zhu, C. F., Shen, H., Hu, J. Z., & Chang, H. X. (2011). Automatic bridge extraction for optical images. In *International conference on image and graphics (ICIG)* (pp. 446–451).

Han, Y., Zheng, H., Cao, Q., & Wang, Y. (2007). An effective method for bridge detection from satellite imagery. In *IEEE conference on industrial electronics and applications* (pp. 2753–2757).

Liu, W., Chena, S. E., Sajedib, A., & Hauserc, E. (2010). The role of terrestrial 3d lidar scan in bridge health monitoring. In *Proceedings of SPIE* (Vol. 7649, pp. 76,491K-1).

Mandlburger, G., Hauer, C., Wieser, M., & Pfeifer, N. (2015). Topo-bathymetric LiDAR for monitoring river morphodynamics and instream habitatsA case study at the Pielach River. *Remote Sensing*, *7*(5), 6160–6195.

Rusu, R. B., & Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *IEEE international conference on robotics and automation (ICRA)* (pp. 1–4).

Sithole, G., & Vosselman, G. (2006). Bridge detection in airborne laser scanner data. *ISPRS Journal of Photogrammetry and Remote Sensing*, *61*(1), 33–46.

Smeeckaert, J., Mallet, C., David, N., Chehata, N., & Ferraz, A. (2013). Large-scale classification of water areas using airborne topographic lidar data. *Remote Sensing of Environment*, *138*, 134–148.