Building Boundary Extraction from LiDAR Point Cloud Data

Abstract—Building boundary extraction from LiDAR point cloud data is important for urban planning and 3D modelling. Due to the uneven point distribution, missing data, and occlusion in LiDAR point cloud data, extraction of boundary points is challenging. Existing approaches have shortcomings either in detecting boundary points on concave shapes or separate identification of "hole" boundary points inside the building roof. In this paper, we present a method for detecting both inner and outer boundary points of the extracted building point cloud. Based on the properties of Delaunay Triangulation and distance from the mean point of the calculated neighbourhood for any point, we extract both inner and outer boundary points. Experimental results using some synthetic shapes as well as some real datasets show the competitive performance of the proposed method.

Index Terms—boundary extraction; building detection; LiDAR data; Performance metric

I. INTRODUCTION

For disaster management, smart city planning, and different geographic information system-based applications, automatic extraction of individual building roofs is important [1], [2]. Using Light Detection And Ranging (LiDAR) data individual building point cloud can be extracted which comprises mainly two parts [3]. The first part mainly segments the input data into ground and non-ground (e.g., buildings and trees) points and then separates trees and other man-made objects from the non-ground points. In the second part, irregular boundaries for each building are extracted followed by a regularisation process. In this paper, we concentrate on the second part and try to extract the accurate boundary points of the individually extracted building.

Methods involving the extraction of individual building outlines from LiDAR point-cloud data can be divided into model-driven and data-driven techniques [9], [10]. The Model-driven approach tries to fit certain boundary shapes previously stored in the database while the data-driven approach tries to find out the exact shape and then regularise the extracted boundary points. The data-driven approaches can be further subdivided into two categories. The first category is the intensity-based approach [4], [5], where, the separated candidate building points are converted into an intensity image. After that, conventional image processing techniques for corners, lines, and edge detection are applied for detecting boundaries from that intensity image. The second category is the direct approach [6], [7]. In this approach, the points are extracted directly based on different geometric properties which represent the exact or approximate boundary points. To form accurate boundaries, regularisation process is performed on the extracted points [3], [8].

Due to the complexities of buildings, different point densities and inner boundaries as in holes, the existing methods sometimes fail to detect all boundary parts correctly. For example Chen et al. [27] can detect boundary points in a simple building but they fail to detect multiple boundary parts of a single complex building separately as demonstrated by Dey et al. [28]. Beside this, extraction of boundary points belong to the inside holes is difficult in some cases [3]. To solve this issue, -shape based algorithms, first introduced by Edelsbrunner et al. [11], have been explored by many authors. The -shape algorithm has the capability to detect both inner and outer boundary points. But it needs to provide the parameter to identify the pairs of points that construct an edge. Defining the value for is not trivial and challenging, specially, when the density of the input point-cloud varies.

In this paper, we propose a data-driven based approach to identify and delineate both inner and outer building boundary points from the separated building point-cloud data.

II. RELATED WORKS

Many indirect data-driven boundary point extraction techniques use binary image masks [3]. Grid of cells is created over the binary image and then boundary is extracted using traditional edge detection algorithms. For example, Awrangjeb et al. [6] considered binary masks constructed from the separated non-ground points and then used canny edge detection algorithm to extract the boundary of individual roof. The binary masks were generated considering different resolutions for different datasets. Similar approaches were used by Verma et al. [13] and Haithcoat et al. [12].

The direct data-driven approaches mainly use different geometric features and parameters to extract boundary contour points [3], [8], [14]. For example, Chen et al. [27] considered the azimuth angles of each point to take decision whether a point is a boundary point or not. Some authors used the convex
hull and/or variation of this algorithm to find out the boundary points from a building cluster [8], [15]. Detecting boundary points of the inner holes is difficult in these cases [3], [28]. For this reason, another group of researchers used a variation of α-shape determination algorithm which has the ability to detect both inner hole and outer boundary points [14], [16], [17].

The main idea of a α-shape algorithm is to use a value for the parameter α to find out the pair of points that form an edge. From the constructed edges, boundary edges are then separated. Andreas et al. [18] choose the average spacing of input point cloud as a value for α and they show that it is a good estimate for building boundary extraction. In the similar way several approaches have been proposed by multiplying the average point spacing by an empirically determined constant for different datasets [19], [20]. However, to define the optimal value for α is challenging. Because, it is directly related to the point density and the density can vary for several reasons including aircraft velocity change, multiple reflection and occlusions [14]. To avoid the problems related to point density variation, Santos et al. [14] choose an adaptive value of α for each edge. They considered the average point spacing of neighbours around the given point of interest. A circle of radius R is considered for choosing neighbours.

To avoid the α-shape algorithm for boundary point extraction, Awrangjeb et al. [3] first formed Delaunay Triangulation (DT) for the individual building point cloud P. Inside the triangulation every side of a triangle is associated with exactly two neighbouring triangles. But, a triangle which is on the periphery of the input point cloud, there is definitely one side which only associated with one triangle. They consider the corresponding pair of points of that side as initial boundary points and traverse all the triangles on the periphery. The problem of using DT in this case is that, some of the edges in the initial boundary can connect the points that are too much far from each other. This incident can happen due to the outlier or irregular LiDAR point distribution. The author considered maximum point-to-point distance (d_{max}) in P and removed those edge which are longer than 2d_{max}. Gilani et al. [7] used the similar approach but they removed the larger edges first and then applied the boundary approximation algorithm of Awrangjeb et al. [3]. The problem of these techniques is that the user must have prior knowledge about the point densities and have to provide the value for d_{max}.

III. PROPOSED METHOD

The main purpose of this paper is to extract both inner and outer boundary points from LiDAR point-cloud data. We compose the proposed method into two major steps: calculation of average point to point distance and identification of appropriate boundary points for individual building.

A. Mean point-to-point distance calculation

Initially, we consider the outer boundary. For this purpose, we calculate the mean point-to-point distance by forming a DT to the input point cloud data P. Initial mean length (l_{initial}) of the edges is calculated considering all edges in the constructed DT. We assume that edges which are at least 3 times greater than l_{initial} are constructed due to the outliers or because of connecting two different parts in P. So, we avoid the longer edges and calculate the final mean length l_{final} of the edges which are less than or equal to 3l_{initial}. Figure 1 shows a demonstration of the process. Fig. 1(a) shows a manually generated A-shape point-cloud data. Delaunay Triangulation is formed in 1(b). Initial edges are coloured by green. Red edges in 1(c) indicate the edges with length greater than or equal to 3l_{initial}.

![Fig. 1. Point-to-point distance calculation. (a) A-shape input point-cloud, (b) constructed Delaunay Triangulation, (c) edges with greater than 3 times than initial mean length.](image)

B. Boundary point identification

Every side of a triangle is associated with two neighbouring triangles except the triangles on the boundary of an object [3]. So, we formed the initial outer boundary using the sides of the triangles which are shared by only single triangle in the constructed DT. However, at this stage, the concavities inside the shapes may not be detected and also we can not separate boundary points of two or more objects (if any) in the point cloud. Thus, to avoid boundary points from two different parts and outliers we remove all edges of length greater than 3l_{initial} and their associative points as boundary. Points associated with the remaining edges shared by only one triangle are considered as boundary points. We denote the extracted boundary point set as β_{a} in this stage.

Boundary points of the inside small holes (if any) may still remain undetected in this stage (as shown in Fig. 3(c)). To solve this issue, we consider k nearest neighbours for each point P_{i} in P. Experimentally, we choose k = 25 for our datasets. Let, the mean point of the calculated k neighbourhood for any point P_{i} is \bar{\mu}. Then, for the cases of non-boundary inner points \bar{\mu} will be much closer to the point P_{i} and the distance d_{m} from \bar{\mu} to P_{i} will be shorter.
For the cases of boundary (both inner and outer) points $d_{m}$ will be larger. We consider this fact and define a threshold for $d_{m}$ based on the method proposed by Tarsha-Kurdi et al. [23]. Fig. 2 demonstrates this situation clearly where, $\bar{\mu}$ (indicated by yellow) represents the mean point of the selected neighbourhood of $P_i$ (red). In the magnified areas, it is clear that for a boundary point, distance $d_{m}$ from $P_i$ to $\bar{\mu}$ is larger than a non-boundary inside point. Based on the calculated distance $d_{m}$ we extract boundary points from $P$ and denote the set of boundary points as $\beta_h$. Finally, we consider $\beta_h \cup \beta_a$ as the set of all inner and outer boundary points.

![Fig. 2. Boundary point detection: distance from mean $\bar{\mu}$ of the neighbouring points to any point $P_i$](image)

Fig. 2. Boundary point detection: distance from mean $\bar{\mu}$ of the neighbouring points to any point $P_i$.

Fig. 3(a) shows the initial edges which only share with one triangles (blue edges). Fig. 3(b) shows the boundary edges after iteratively removing the boundary edges greater than $3 \times L_{\text{initial}}$. Inner edges are shown in Fig. 3(c) (purple lines). Fig. 3(d) shows both inner and outer extracted boundary points using the proposed method.

![Fig. 3. Boundary identification with inner hole: (a) initial edges shared with single triangle, (b) extracted outer boundary edges (green lines), (c) edges of inner small hole (purple lines), (d) extracted inner and outer boundary edge points (red points)](image)

Fig. 3. Boundary identification with inner hole: (a) initial edges shared with single triangle, (b) extracted outer boundary edges (green lines), (c) edges of inner small hole (purple lines), (d) extracted inner and outer boundary edge points (red points)

IV. EXPERIMENTAL RESULTS

To test the proposed method, we first manually generated some shapes using real point-cloud data. We find the performance of the detected boundaries with respect to the ideal regular boundary of the corresponding shapes. After that, we applied and evaluated the proposed method to real datasets for building boundary point extraction.

A. Datasets

Three shapes (A, E and M) are generated manually from real point cloud data with different point densities. Reference shapes are also generated as shown in Fig. 4, where, left side shows M-shape, A-shape and E-shape point cloud, respectively, and right side indicates their corresponding reference shapes with point-cloud.

To apply the proposed method in real building point cloud data, we choose three different datasets. First one is Australian benchmark datasets [22], second one is from ISPRS benchmark [24] and third datasets is a selected site from Strasbourg (SB) city of France [29]. Two different areas from Australian datasets are selected. One is from Aitkenvale (AV) area with point density 40 points/m² and other one is from Hervey bay (HB) area with a 12 points/m² density. From ISPRS benchmark we selected a site from Vaihingen (VH) area with a point density of 2.5 to 3.9 points/m². Fig. 5(a) and 5(c) show the selected sites of Australian datasets and 5(b) shows the site of ISPRS benchmark datasets with the referenced building boundaries. Both of the datasets contain buildings from residential areas. The Strasbourg datasets contain several buildings with large and small inside holes. Thus, to show the performances of inner and outer boundary point extraction using our proposed method, we selected some buildings from that datasets containing holes or spaces inside the buildings. The point density of this datasets is low (1.3 points/m²). Using the methods proposed by Dey et al. [25] we extract the individual building area and then detect the boundary points using the proposed method.

![Fig. 4. Generated shapes. Left column indicates the generated shapes with different point-densities and right side indicates corresponding regular reference shapes](image)
Fig. 5. Sites of the used datasets with referenced building boundary (cyan polygons): (a) Aitkenvale area (AV), (b) Vaihinen area (VH), (c) Hervey bay site (HB) and (d) Strasbourg site (SB).

B. Results

We compare the proposed method with some existing state-of-the-arts boundary point extraction techniques proposed by Awrangjeb et al. [3], Dey et al. [25] and Santos et al. [14], respectively. To evaluate the performances, we consider three different metrics. Firstly, the polygon and line segments (PoLiS) distance proposed by Avbelj et al. [21], secondly, the Robust Corner Correspondence (RCC) metric proposed by Dey et al. [26] and thirdly, the constructed area by the detected boundary points. For real datasets, completeness ($C_m$), correctness ($C_r$) and quality ($Q_l$) of the detected building boundaries are also considered.

For PoLiS metric, distances from each extracted boundary point to the closest point of reference boundary is calculated. Mean of the distances is considered as final PoLiS distance for each building. For RCC, one to one true corner correspondences between the extracted and reference building boundaries are calculated first and then mean distance is calculated for the extracted boundary points. To find the total area inside the boundary we have used Matlab built-in functions. Ideally, the mean distance should be close to zero for a perfect detection. So, smaller distance is expected. Calculated area inside the detected boundary points should be close enough to the area of the reference shape. Consider the Fig. 6, where green points indicate the detected boundary points and individual red arrow in Fig 6(a) indicate the distance from a detected boundary point to the reference. Fig. 6(b) demonstrates an example of the detected area surrounded by red line and actual area surrounded by blue lines.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.550</td>
<td>0.572</td>
<td>0.530</td>
<td>0.598</td>
</tr>
<tr>
<td>E</td>
<td>0.057</td>
<td>0.061</td>
<td>0.059</td>
<td>0.052</td>
</tr>
<tr>
<td>M</td>
<td>0.125</td>
<td>0.120</td>
<td>0.126</td>
<td>0.130</td>
</tr>
</tbody>
</table>

Table I shows the calculated PoLiS distances from corresponding reference boundary to the detected boundary points for different shapes. We indicate the methods proposed by Awrangjeb et al. [3], Dey et al. [25] and Santos et al. [14] as MA, EKD and DOS, respectively. From this table it is clear that proposed method shows less distances most of the cases which is expected. Again, Table II shows the calculated boundary areas of different shapes using different methods. In all 3 cases, areas of detected boundaries using the proposed method are close to the actual reference boundary area, which is also expected.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>460.82</td>
<td>470.01</td>
<td>483.62</td>
<td>488.81</td>
<td>560</td>
</tr>
<tr>
<td>E</td>
<td>1114.51</td>
<td>1149.65</td>
<td>1144.32</td>
<td>1159.21</td>
<td>1280</td>
</tr>
<tr>
<td>M</td>
<td>1369.8</td>
<td>1425.9</td>
<td>1512.2</td>
<td>1502.83</td>
<td>1575</td>
</tr>
</tbody>
</table>

Table III compares the calculated distances from the detected boundary points to the reference boundary using different methods for the real datasets and Fig.7 shows the qualitative results of the proposed boundary point detection on Aitkenvale area as a representative of real datasets. Fig. 8 shows the capabilities of the proposed method in terms of detecting both inner and outer boundary points in real datasets using some selected buildings from Starsbourg area. Using Table IV we show the quantitative performances in terms of completeness ($C_m$), correctness ($C_r$) and quality ($Q_l$) of the proposed method along with two other state-of-the-arts techniques for our used datasets.
### TABLE III
CALCULATED DISTANCES (IN METER) USING PoLiS AND RCC METRICS FOR DIFFERENT METHODS USING REAL DATASETS

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PoLiS RCC</td>
<td>0.18 0.26</td>
<td>0.21 0.22</td>
<td>0.18 0.25</td>
<td>0.22 0.24</td>
</tr>
<tr>
<td>HB</td>
<td>0.59 0.42</td>
<td>0.53 0.48</td>
<td>0.62 0.55</td>
<td>0.58 0.51</td>
</tr>
<tr>
<td>VH</td>
<td>0.38 0.41</td>
<td>0.33 0.39</td>
<td>0.31 0.42</td>
<td>0.31 0.32</td>
</tr>
</tbody>
</table>

All the distances are calculated in meters.

---

### TABLE IV
EVALUATION OF PIXEL-BASED BOUNDARY EXTRACTION IN AUSTRALIAN BENCHMARK

<table>
<thead>
<tr>
<th>Method</th>
<th>$C_m$</th>
<th>$C_r$</th>
<th>$Q_I$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avrangjeb et al. [22]</td>
<td>0.85</td>
<td>0.80</td>
<td>0.71</td>
</tr>
<tr>
<td>Dey et al. [25]</td>
<td>0.83</td>
<td>0.97</td>
<td>0.86</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.89</td>
<td>1.00</td>
<td>0.97</td>
</tr>
<tr>
<td>Avrangjeb et al. [22]</td>
<td>0.85</td>
<td>0.72</td>
<td>0.64</td>
</tr>
<tr>
<td>Dey et al. [25]</td>
<td>0.85</td>
<td>0.85</td>
<td>0.83</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.93</td>
<td>0.98</td>
<td>0.99</td>
</tr>
<tr>
<td>Avrangjeb et al. [22]</td>
<td>0.82</td>
<td>0.89</td>
<td>0.73</td>
</tr>
<tr>
<td>Dey et al. [25]</td>
<td>0.79</td>
<td>1.00</td>
<td>0.78</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.88</td>
<td>1.00</td>
<td>0.96</td>
</tr>
<tr>
<td>Avrangjeb et al. [22]</td>
<td>0.84</td>
<td>0.90</td>
<td>0.79</td>
</tr>
<tr>
<td>Dey et al. [25]</td>
<td>0.85</td>
<td>1.00</td>
<td>0.80</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.92</td>
<td>1.00</td>
<td>0.97</td>
</tr>
</tbody>
</table>

---

**V. Conclusion**

We propose a method for detecting boundary point from point cloud data. Based on the Delaunay Triangulation, we find the average point spacing in the point cloud of individually extracted building. By using the traditional $k$-nearest neighbouring algorithm, we find the neighbourhood for each point and then mean point of the selected neighbourhood is calculated. We use the properties of Delaunay Triangulation and distance from the mean point of the calculated neighbourhood to select outer and inner boundary, respectively. We show the competitive performances using some synthetic shapes and also for real datasets with complex buildings with holes, convex or concave segments. The performances of the proposed method may degrade in the cases of abrupt point densities in the extracted building point cloud data. Moreover, the selection of proper value for $k$ is challenging. In future, we will try to solve these issues.

---

**REFERENCES**


