remote sensing

MDPI

Article

Edge Detection and Feature Line Tracing in 3D-Point Clouds by Analyzing Geometric Properties of Neighborhoods

Huan Ni [1], Xiangguo Lin [2], Xiaogang Ning [2] and Jixian Zhang [1,*]

[1] School of Resource and Environmental Sciences, Wuhan University, No. 129 Luoyu Road, Wuhan 430079, China; nih2015@yeah.net

[2] Chinese Academy of Surveying and Mapping, No. 28 Lianhuachixi Road, Beijing 100830, China; linxiangguo@gmail.com (X.L.); ningxg@casm.ac.cn (X.N.)

* Correspondence: zhangjx@casm.ac.cn; Tel.: +86-10-6388-1816

Academic Editors: Jie Shan, Juha Hyyppä, Gonzalo Pajares Martinsanz and Prasad S. Thenkabail
Received: 10 March 2016; Accepted: 24 August 2016; Published: 1 September 2016

Abstract: This paper presents an automated and effective method for detecting 3D edges and tracing feature lines from 3D-point clouds. This method is named Analysis of Geometric Properties of Neighborhoods (AGPN), and it includes two main steps: edge detection and feature line tracing. In the edge detection step, AGPN analyzes geometric properties of each query point's neighborhood, and then combines RANdom SAmple Consensus (RANSAC) and angular gap metric to detect edges. In the feature line tracing step, feature lines are traced by a hybrid method based on region growing and model fitting in the detected edges. Our approach is experimentally validated on complex man-made objects and large-scale urban scenes with millions of points. Comparative studies with state-of-the-art methods demonstrate that our method obtains a promising, reliable, and high performance in detecting edges and tracing feature lines in 3D-point clouds. Moreover, AGPN is insensitive to the point density of the input data.

Keywords: 3D edge; Edge detection; Feature line tracing; RANdom SAmple Consensus (RANSAC); Angular gap

1. Introduction

1.1. Problem Statement

Feature extraction in 2D-images, one of the most important topics in the fields of image analysis and computer vision, has been studied for years [1]. Edges and feature lines are considered as important features in various urban scenes covering a vast number of man-made objects. Generally, once edges are detected, a further step will be done for tracing feature lines from the detected edges. For edge detection in images, edges have been well defined, such as "the boundary element between two regions of different homogeneous luminance" [2] or "large or sudden changes in some image attribute, usually the brightness" [3]. An extensive review of the established edge detection methods can be found in the literature [1]. Apart from the surveyed methods, many outstanding methods have been proposed such as the revised Canny operator [4] and Edison operator [5].

In recent years, benefiting from the advances in sensor technology for both airborne and ground-based laser scanning, dense 3D-point clouds have become increasingly common [6]. Therefore, edge detection and feature line extraction in 3D-point clouds have become a novel research topic. However, in the fields of remote sensing and photogrammetry, the approaches have just scratched the surface, and the definition of 3D edges has not yet been confirmed though feature line extraction has long been a major issue in the 3D city modeling works.

In addition, the established edge detection methods defined for images cannot be applied directly to 3D-point clouds. The main reasons for this are given below:

(1) The data representation is different. An image is considered as a matrix, whereas a 3D-point cloud is an unorganized and irregularly distributed [7] scattered point set.
(2) The presented information type is different. An image contains cryptic spatial information, and abundant spectral information. Comparatively, a 3D-point cloud contains explicit spatial information, and the reflected intensity at times [8].
(3) The spatial neighborhood is different. An image is arranged as a grid-like pattern, and the neighborhood of a pixel can easily be determined. However, a 3D-point cloud is unorganized, and the neighborhood of a point is more complex than that of a pixel in an image. Generally, in 3D-point clouds, there are three types of neighborhoods: spherical neighborhood, cylindrical neighborhood, and k-closest neighbors based neighborhood [9]. The three types of neighborhoods are based on different search methods, and change of the search method alters the neighborhood correspondingly.

To address the aforementioned problems, an automated and effective method is proposed to detect edges and trace feature lines from 3D-point clouds. Prior to presenting our method, the definition of 3D edges is given herein. Traditionally, 2D edges in an image are defined as the following two types [10]:

(1) Gray level edges, which are often associated with abrupt changes in average gray level.
(2) Texture edges, which are the abrupt "coarseness" changes between adjacent regions contained the same texture at different scales, or the abrupt "directionality" changes between the directional textures in adjacent regions.

Then, the definition of 2D edges is extended by the literatures [1–4,11]. Specially, the literature [1] defines 2D edges as one-dimensional discontinuities in the intensity surface of the underlying scene. However, the intensity or spectral information cannot describe the complete geometric properties in 3D-point clouds. According to the definitions in images and the characteristics of 3D-point clouds, we visibly define 3D edges as 3D discontinuities of the geometric properties in the underlying 3D-scene.

Mathematically, we define 3D edges as the following two types (see Figure 1):

(1) Boundary elements, which are often associate with an abrupt angular gap in the shape formed by their neighborhoods. The details are presented in Section 2.2. Boundary elements are the edges belonging to roof contours, façade outlines, height jump lines [12], and other types of surface's contours. Specially, the surface is a 3D-plane or a curve surface.
(2) Fold edges, which are the abrupt "directionality" changes between the normal directions in adjacent surfaces. Generally, two curve or planar intersected surfaces exist in the neighborhood of a fold edge. The details are presented in Section 2.2. Fold edges are the edges belonging to plane intersection lines [13], sharp feature line [14], breaklines [15], and other types of intersections between different surfaces.

Edge detection in 3D-point clouds is similar to 2D image processing. The usual aim of edge detection is to locate edges belonging to boundaries of objects of interest [3]. Most edge detection techniques consist of two stages [11]: (1) converting an original image into features; and (2) assigning points to edges or non-edges. Similarly, our proposed edge detection procedure first computes angular gap feature for all the points in a 3D-point cloud, then assigns the points to edges or non-edges. Sometimes, there is a further stage called edge linking, in which the detected edges are examined once again for instance to obtain closed contours [11]. For 3D-point clouds, the edge linking procedure may relate to a special application such as line segment extraction or building reconstruction. In this paper, we present a feature line tracing procedure for linking the detected edges to feature lines, such as boundaries and intersection lines. The feature lines might be straight or curve, however, must be smooth.

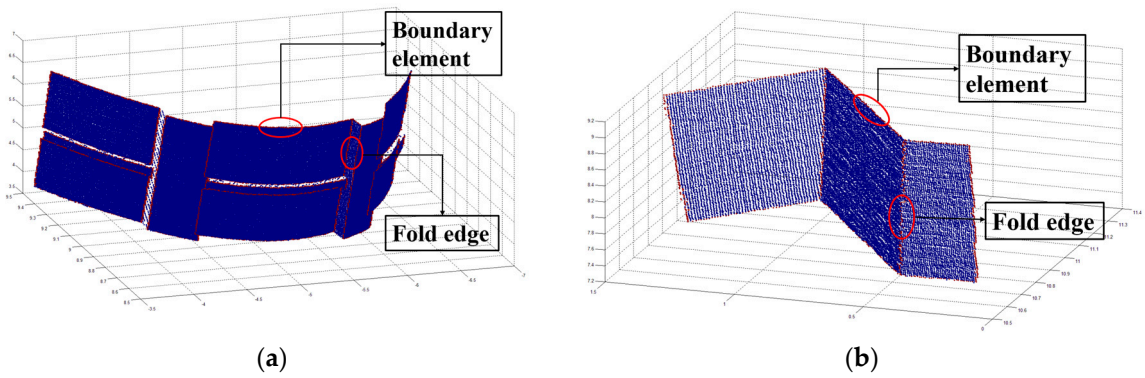


Figure 1. The definition of two types of edges.

To detect the defined 3D edges and trace the feature lines from 3D-point clouds, we propose an Analysis of Geometric Properties of Neighborhoods (AGPN) method. AGPN first analyzes geometric properties of each query point's neighborhood in spatial domain, and then detects 3D edges based on RANSAC and angular gap metric. Finally, to trace feature lines from the 3D edges, a hybrid method based on region growing and model fitting is proposed. With the proposed AGPN method, we can detect the two defined 3D edges—boundary elements and fold edges from 3D-point clouds directly without extra image processing or point cloud preprocessing (e.g., segmentation or object recognition).

1.2. Related Work

In the fields of remote sensing, photogrammetry, and computer vision, edge detection and feature line extraction from airborne or terrestrial laser scanning data, has long been one of the major issues. Specifically, feature line extraction from 3D-point clouds, as a substep in 3D city modeling works, has been a major research topic for years [12,13,15–35]. Research efforts for 3D edge detection and feature line extraction from laser scanning data can be categorized into two groups: (1) direct methods [12,13,15–26], first recognize the building points from a 3D-point cloud, next, segment or cluster building points into planes, finally, detect edges and extract feature lines i.e., plane outlines and plane intersection lines [13]; (2) indirect methods [26–35], first convert a 3D-point cloud into an image, or register with corresponding images, next, detect 2D edges in the images, and then project the 2D edges back into the 3D-point cloud to obtain 3D edges.

The above-noted approaches can detect edges on regular feature lines such as plane intersection lines [13] or breaklines [15,16], roof or wall outlines [15,17] in buildings or piecewise planar objects. We mainly review the aforementioned direct methods herein, because the indirect methods might cause loss of spatial geometric information when a 3D-point cloud is converted into an image. To extract roof boundaries or façade outlines, a boundary estimation method is required in the direct methods. At beginning, 3D-point clouds without a high point density bring challenges to this task [18]. The literature [12] extracts roof boundaries named height jump lines by using segmented ground plans. With the improvement of the point density of 3D-point clouds, two groups of boundary estimation methods are proposed. The first group of methods [17,19,20] extract boundaries based on triangulation. In this case, a Triangular Irregular Network (TIN) for the planar segments is generated first, and then long TIN edges appear only at the outer boundary (segment outlines) or inner boundary (holes). Boundary points are just the end points of the long TIN edges. The second group of methods [21,22] extract boundaries based on convex hull algorithm. In this case, the convex hull algorithm is modified by local convex hull testing to deal with complex boundaries. Then the points which are close enough to the query local convex boundary are picked up and treated as boundary points. In comparison with boundary extraction, the extraction for plane intersection lines is more convenient in the field of photogrammetry. After all the planar segments in a 3D-point cloud have been determined, the topologic relations among the planar segments are computed and represented by an adjacency matrix [15], then

all pairs of adjacent planar segments are intersected to extract plane intersection lines [12,13,15,23], and thus, the edges on the intersection lines can be easily detected.

The aforementioned methods can, however, detect edges and extract feature lines in only regular buildings or piecewise planar objects. Furthermore, the literature [6] reviews the drawbacks of this work in the literature [13], some of which are common in state-of-the-art methods using 3D laser scanning data. The drawbacks include: (1) these methods are incapable of fitting a small and narrow plane in a noisy point cloud; (2) these methods may generate unexpected lines at non-planar surfaces when the data become complex.

Some approaches [14,36–40] employ surface meshes or point-based surfaces, which can detect 3D edges or extract feature lines from some irregular objects and more complex surfaces. However, these methods are only applied to a small-scale 3D-point cloud with a single object. Specially, an angular gap based method [39,40], whose variations are widely used in building reconstruction work [41], fan clouds work [42] and the Point Cloud Library (PCL) [43], has been proposed. In this paper, the angular gap based method is utilized as a criterion in our proposed method.

2. Methodology

2.1. Overview

To detect 3D edges and trace feature lines, a two-step strategy based on AGPN, is illustrated in Figure 2. In the diagram shown in Figure 2, the upper part marked by blue dotted lines is the first step, and the lower part marked by red dotted lines is the second step. The first step, detailed in Section 2.2, detects or locates edges in 3D-point clouds based on RANSAC, normal optimization, and angular gap computation. The second step, detailed in Section 2.3, traces feature lines from the detected edges based on neighborhood refinement and growing criterion determination.

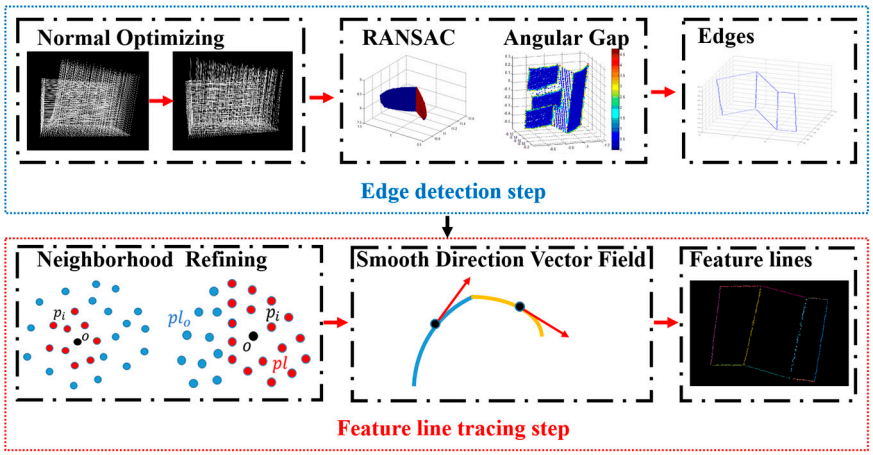


Figure 2. Overview of the proposed AGPN.

The pseudo code of our proposed AGPN method and its parameters are shown in Appendix A.

2.2. Edge Detecton

2.2.1. Geometric Property Analysis

The inherent property of being an edge is that it is the local neighborhood of a point rather than the point itself [44]. Based on this principle, we design an algorithm to determine whether a point is an edge or not by analyzing geometric properties of the point's neighborhood. In the neighborhood of a boundary element, there is only one curve or planar surface. In the neighborhood of a fold edge,

there are two or more intersected surfaces. We first present the flowchart of the edge detection step, and then explain why the defined edges can be detected.

The flowchart of the edge detection step in our proposed AGPN is shown in Figure 3. Let o denote an unlabeled point. Our method first searches the nearest neighbor point set P of o based on distance. Next, the point set P is fitted into a local plane, pl, by the RANSAC algorithm, and then P is divided into inliers (on the fitted plane pl) and outliers. The point o will be labeled as non-edge if it does not belong to the inliers. Otherwise, o and the inliers are connected to construct a number of spatial vectors, from which angular gap will be calculated based on the optimized normal detailed in Section 2.2.4. If a substantial angular gap exists between the constructed spatial vectors on pl, o will be labeled as edge. Otherwise, o will be labeled as non-edge. Edges will be detected after all the points in a 3D-point cloud are labeled.

It is noteworthy that we use the angular gap method rather than a modified convex hull boundary detection algorithm, which can be widely found in some references, because the inliers may be a subset of a surface plane model. Moreover, a *kd*-tree is used to determine the nearest neighbor point set P of each point.

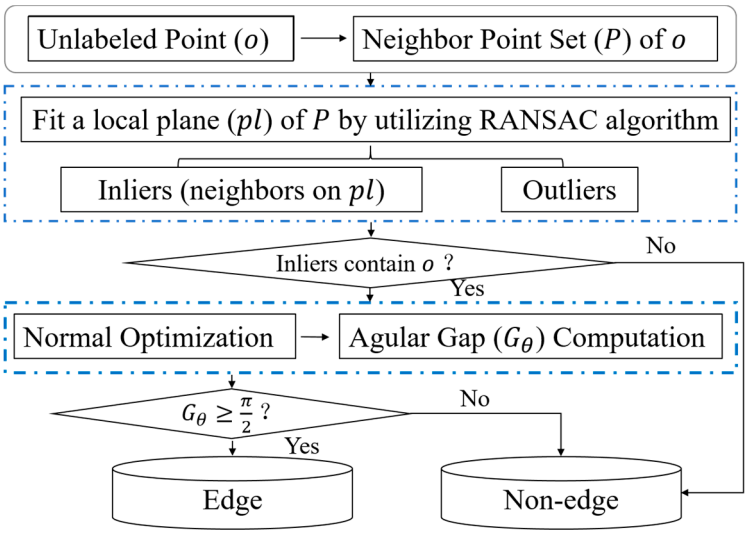


Figure 3. Flowchart for the edge detection step in AGPN.

As shown in Figure 3, the plane model, calculated by the RANSAC algorithm, fits a local surface in the nearest neighbor point set P of the unlabeled point o (see Figure 4). When o is on a plane outline or plane intersection line, the plane model is impeccable. When o is on a curve surface boundary or intersection of different curve surfaces, the plane model is also the most direct and effective geometric model for approximating the local smooth surface, because the neighborhood of o is a local small area of the surface.

Based on the plane model fitted by the RANSAC algorithm, the nearest neighbor point set P are divided into inliers and outliers (see Figure 4). The inliers are in the fitted plane and the outliers are outside. It can be found that, by the RANSAC algorithm, a best plane model can be found in P. If a point is a boundary element, the inliers are on the fitted plane, and the outliers are noise. If a point is a fold edge, the inliers are on the fitted plane lying on one of the intersecting surfaces in the point's neighborhood (see Figure 4).

The edge detection procedure in AGPN can detect both boundary elements and fold edges. The detection of the two types of edges is detailed in Sections 2.2.2 and 2.2.3, respectively.

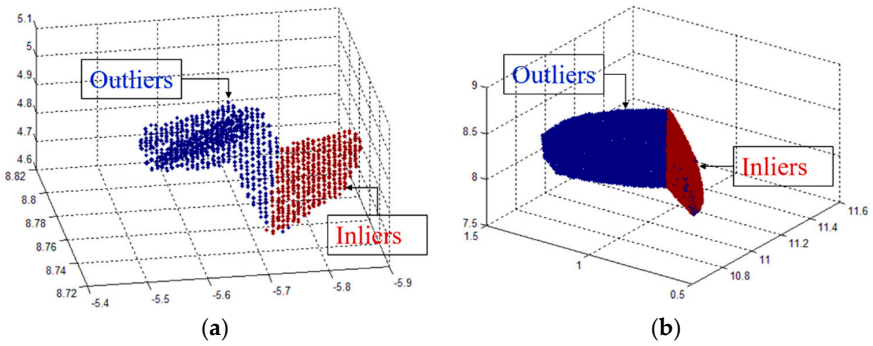


Figure 4. Local plane (rendered in red) fitted by the RANSAC algorithm in the nearest neighbor point set P. (**a**,**b**) show two types of neighbor point sets respectively. There are three planes in (**a**) and two planes in (**b**).

2.2.2. Boundary Element Detection

As shown in Figure 5, there is only one surface in the neighborhood of an unlabeled point o. K points depicted in blue and red are the nearest neighbors of o, obtained using a *kd*-tree. Points $p_i \ (i = 1 \cdots N_r)$ rendered in red are inliers extracted by the RANSAC algorithm. These inliers are on the fitted plane pl of the local surface.

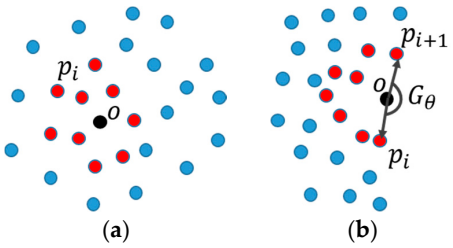


Figure 5. Distribution of the nearest neighbors of an unlabeled point on a surface: (**a**) the neighborhood of an interior point; (**b**) the neighborhood of a point on a boundary.

If o is on a surface boundary, there will be a substantial angular gap G_θ (see Figure 5b) between vectors $\overrightarrow{op_i} \ (i = 1 \cdots N_r)$ on the local plane pl. If o is an interior point (see Figure 5a), the distribution of the angles between vectors $\overrightarrow{op_i} \ (i = 1 \cdots N_r)$ will be consecutive, and there will certainly be no substantial angular gap.

Notably, o will be labelled as noise or an isolated point if it is an outlier of the local plane pl.

2.2.3. Fold Edge Detection

As shown in Figure 6, the neighborhood of an unlabeled point o includes two intersecting surfaces. K points depicted in blue and red are the nearest neighbors of o. Points $p_i \ (i = 1 \cdots N_r)$ rendered in red are inliers extracted by the RANSAC algorithm. The extracted inliers are on the fitted local plane pl lying on one of the intersecting surfaces.

If o is a fold edge point, there will be a substantial angular gap G_θ (see Figure 6b) between vectors $\overrightarrow{op_i} \ (i = 1 \cdots N_r)$ on the local plane pl. The local plane pl is on one of the intersecting surfaces. If o is an interior point on one of the intersecting surfaces (see Figure 6a), the distribution of the angles between vectors $\overrightarrow{op_i} \ (i = 1 \cdots N_r)$ will be consecutive, and there will certainly be no substantial angular gap.

Figure 6c shows a special case, that is, the point densities of the intersecting surfaces are considerably different. One of the intersecting surfaces contains sparse points, while the other intersecting surface includes much dense points. If o is an interior point on the surface with sparse points, the inliers extracted by the RANSAC algorithm will be on the other surface with dense points.

Then, once the angular gaps are computed from these inliers, there will be a substantial angular gap, resulting in o mislabeled as an edge. Fortunately, o is an outlier of the fitted local plane pl, and therefore, it can be rejected by determining whether or not it is an inlier of the fitted local plane pl.

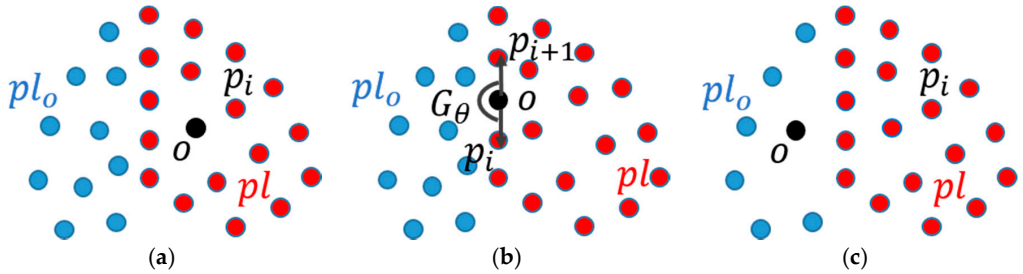


Figure 6. Distribution of the nearest neighbors of an unlabeled point on a surface intersecting structure: (**a**) the neighborhood of an interior point on one of the intersecting surfaces; (**b**) the neighborhood of a fold edge point; (**c**) the neighborhood of a point on the two intersecting surfaces with different point densities.

In addition, if the number of inliers is less than three, the inliers will not be able to fit a plane. In this case, o will be labeled as noise or a local extreme, such as the vertex of a circular cone.

2.2.4. Normal Optimization

Generally, the normal of a 3D point is computed by a covariance matrix created from the nearest neighbors of the 3D point [45], which is called PCA-Normal herein. The PCA-Normal is the normal of the tangent plane of a 3D point. In our method, the normal is required to be orthogonal to the local fitted plane pl. When we detect boundary elements, only one surface exists in the neighborhood, and pl is on the surface. When we detect fold edges, two or more intersecting surfaces exist, and pl is on one of the intersecting surfaces. However, the PCA-Normal is orthogonal to the tangent plane, and hence it cannot meet the aforementioned requirement.

The reasons for this are that the PCA algorithm cannot detect any of the intersected planes in a complex neighborhood. Fortunately, the RANSAC algorithm can solve this problem, and deal with noise as well. As shown in Figure 4, the RANSAC algorithm can detect a proper plane in either of the two complex neighborhoods.

In this study, we fit a local plane in the neighborhood to estimate the normal $\vec{n}$ of an unlabeled point o. The procedure first searches for the neighbors of o. Next, to eliminate the influence of outliers or noises, a local plane pl is fitted using the RANSAC algorithm. Then, $\vec{n}$ is the normal of the local fitted plane pl. Note that the normal $\vec{n}$ used in this study is named RANSAC-Normal.

2.2.5. Angular Gap Computation

To compute the angular gap, G_θ, a spatial coordinate frame is constructed based on the local plane pl and its normal vector $\vec{n}$. The axes of this frame are composed of two perpendicular vectors $\vec{u}$, $\vec{v}$ on pl and the normal vector $\vec{n}$. The angular gap G_θ is computed by Equations (1)–(4):

$$d_i^u = \vec{op_i} \odot \vec{u} \tag{1}$$

$$d_i^v = \vec{op_i} \odot \vec{v} \tag{2}$$

$$\theta_i = arc\tan\left(d_i^u / d_i^v\right) \tag{3}$$

$$G_\theta = \max\left(\theta_{i+1} - \theta_i\right) \ \ (i = 1 \cdots N_r - 1) \tag{4}$$

where $\vec{op_i}$ $(i = 1 \cdots N_r)$ is the vector connecting the unlabeled point o to an inlier p_i extracted by the RANSAC algorithm. The distribution of G_θ is shown in Figure 7.

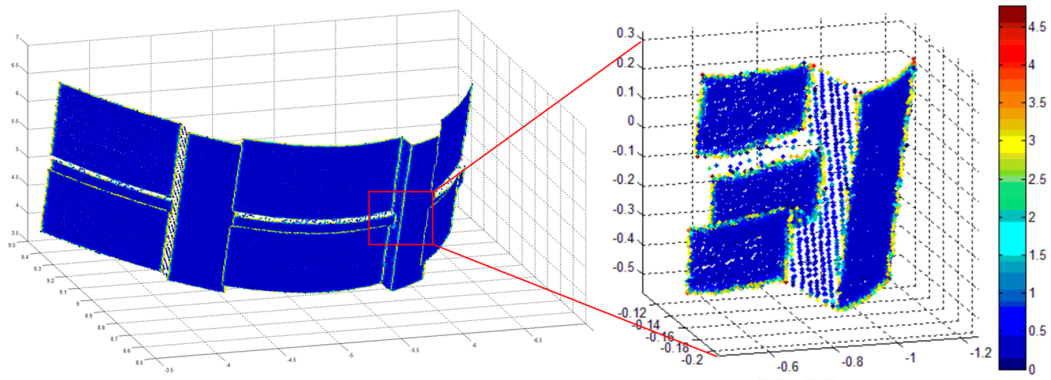


Figure 7. Distribution of G_θ , each point is colored according to its value of G_θ .

The pseudo code of the edge detection step in AGPN and its parameters are shown in Appendix B.

2.3. Feature Line Tracing

Once edges have been detected, a further step will trace the detected edges into segments. Each segment is a point list, in which all the points belong to the same feature line. The proposed feature line tracing method connects edges with similar principle directions and splits edges with abrupt directionality changes. Thus, the traced feature lines are curve or straight, however, must be smooth.

The proposed feature line tracing is a hybrid method based on region growing and model-fitting algorithms. The model-fitting algorithm estimates 3D line parameters in each point's neighborhood by the RANSAC algorithm. The directional parameters of the fitted 3D line denote the principle direction of the edge point. The region growing algorithm clusters the detected edges into segments based on the following two redefined growing criteria related to the refined neighborhood and the principle direction.

The proposed feature line tracing procedure includes two essential steps: neighborhood refinement and growing criterion definition.

2.3.1. Neighborhood Refinement

Compared to an image, a 3D-point cloud is an unorganized and scattered point set, which brings great challenges to neighborhood searching of edge points. Our method first obtains the nearest neighbors of a query point by using the *kd*-tree algorithm. Next, a straight line model is fitted by the RANSAC algorithm, and then, the nearest neighbors are divided into inliers and outliers. The inliers containing the query point are the refined nearest neighbors. Otherwise, the outliers are processed iteratively by the RANSAC algorithm until the updated inliers contain the query point. Therefore, an adaptive neighborhood is designed for each query point.

2.3.2. Growing Criterion Definition

Two growing criteria given by [46] are redefined in this study.

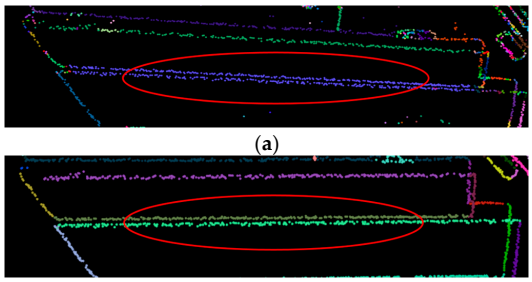
a Proximity of points. Only points that are near one of the points in the current segment can be added to the stack of the segment. For feature line tracing, this proximity of edge points can be implemented by the aforementioned neighborhood refinement.

b Smooth direction vector field. Only points that have a similar principal direction with the current tracing segment can be added to the stack of the current segment. In this paper, a line model is first fitted from the refined neighborhood by the RANSAC algorithm, and then the principal direction of the current point is defined as the direction of the fitted line.

In addition, a larger proportion of inliers to all nearest neighbors implies a higher possibility of the presentence of feature lines. Due to the region growing procedure being irreversible, an edge point

with greater linearity should be grown first to ensure a better tracing result. Therefore, edge points are first sorted by their proportion values, and then grown in order.

The general region growing procedure for linear feature segmentation is sensitive to the size of the established local neighborhoods and the location of seed points. However, the refined neighborhood and the aforementioned sorted edge points can overcome the problem. It is validated that the proposed feature line tracing procedure can distinguish spatially-adjacent, collinear/coaxial lines (see Figure 8).



(**a**)

(**b**)

Figure 8. Feature line tracing, (**a**) feature line segments generated by region growing method; (**b**) feature line segments generated by the proposed feature line tracing method. The traced segments are marked by different colors.

Three parameters are used in the feature line tracing step, that is, the number of nearest neighbors K^2 for *kd*-tree algorithm, distance threshold d_r^2 for the RANSAC line model estimation, and smooth direction threshold *sm_thr*. The traced feature lines are shown in Figure 8b.

3. Experiments and Analysis

The proposed algorithms were implemented in C++ using the PCL. There are five parameters in the proposed AGPN (detailed in Appendix A). In this study, the point spacing of the input data affects the performance of our proposed AGPN most. Moreover, the performance is also affected by the distance thresholds d_r^1 and d_r^2 related to the point spacing. Therefore, we describe the testing data in Section 3.1 and discuss parameters tuning in Section 3.3. The point spacing is measured by the open source software CloudCompare [47].

To quantitatively evaluate the performance of our AGPN, four measures are defined in Section 3.2. We further compare the proposed AGPN with two representative algorithms: the boundary estimation method presented in the PCL and the edge points clustering algorithm presented in the literature [13]. The comparative studies are presented in Section 3.7.

3.1. Testing Data

The open datasets available from the homepage of 3D ToolKit [48] are employed. The datasets are recorded by a Riegl VZ400 scanner in the Bremen city center. The point density of a single 3D scan is uneven.

From the open datasets, we selected two testing sites, i.e., Site 1 and Site 2. The two sites contain a large number of complex man-made objects. Table 1 shows their detailed information, including the number of points, maximum point spacing, minimum point spacing and the parameters used in our experiments.

The point spacings in the two testing sites are quite different. In Site 1, the point density is uneven. In the neighborhood of the scanner, the average point spacing is 0.001 m, and in the areas away from the scanner, the average point spacing is 0.15 m. The variation tendency of the point density is consecutive with the variation of the distances to the scanner. In Site 2, except for the window areas, the average point spacing is 0.005 m. In the window areas, the point density decreases rapidly, and the average point spacing turns to 0.01 m. The variation tendency of the point density is piecewise.

Table 1. Data description and parameter settings for the two testing sites.

	Number of Points	Maximum Point Spacing	Minimum Point Spacing	Parameters				
				K^1	d_r^1	K^2	d_r^2	*sm_thr*
Site 1	14040449	0.15	0.001	200	0.01	15	0.01	0.2
Site 2	4411599	0.01	0.005	200	0.005	15	0.005	0.2

3.2. Evaluation Metrics

To test the performance of the proposed AGPN, we quantitatively evaluate the results of the edge detection step and the feature line tracing step, respectively, by four measures: p_{dc}(correctness rate of the edge detection step), p_{mj}(mislabeled rate of the edge detection step), p_{dct}(correctness rate of the feature line tracing step), and p_{mjt}(mislabeled rate of the feature line tracing step). To compute p_{dc} and p_{mj} for evaluating the 3D edge results, we count the number of the feature lines which the detected edges belong to. The feature line is curve or straight, however, must be smooth. The four measures are defined as follows:

$$p_{dc} = \frac{N_{dc}}{N_{gc}} \tag{5}$$

$$p_{mj} = \frac{N_{mj}}{N_{gc}} \tag{6}$$

$$p_{dct} = \frac{N_{tc}}{N_{dc} + N_{mj}} \tag{7}$$

$$p_{mjt} = \begin{cases} \frac{N_{mjt}}{N_{dc}+N_{mj}} \; if \; N_{mjt} \leq N_{dc} + N_{mj} \\ \frac{N_{mjt}}{N_{mjt}+N_{tc}} \; Otherwise \end{cases} \tag{8}$$

where N_{dc} is the number of true positive feature lines contained in the detected edges, N_{tc} is the number of correctly traced feature lines in the feature line tracing step, N_{gc} is the total number of feature lines in the ground truth, N_{mj} is the number of mislabeled feature lines contained in the detected edges, and N_{mjt} is the number of incorrectly traced feature lines in the feature line tracing step. Larger values of p_{dc} and p_{dct}, and smaller values of p_{mj} and p_{mjt} are more desirable.

3.3. Parameter Tuning

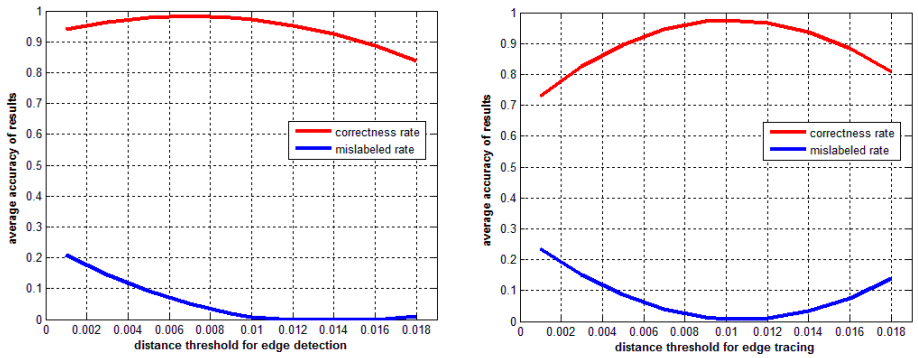
One of the major strengths of the proposed AGPN is that the extracted 3D edges in the edge detection step are sufficiently subtle with the default value ($\frac{\pi}{2}$) of the angular gap (G_θ).

Table 1 lists the parameters used in the proposed AGPN and their empirical values. We maintained K^1 = 200 and K^2 = 15 for Site 1 and Site 2 because both K^1 and K^2 have little influence on the quality of the results.

In addition, because the feature line tracing parameters are mainly dependent on the requirements of users, we only analyze the influence on the application of 3D line segment extraction. Under this condition, to ensure the sufficient quality of the result, *sm_thr* is set to 0.2.

We selected a number of subsets in Site 1 with the same point spacing (0.01 m) to analyze the sensitivity of d_r^1 and d_r^2. As shown in Figure 9, the x-axis denotes d_r^1 and d_r^2, the y-axis depicts the average values of the correctness and mislabeled rates. It can be seen that when the value of d_r^1 and d_r^2 are close to the point spacing of 0.01 m, the edge detection step and the feature line tracing step both achieve good results. If we set d_r^1 smaller than the point spacing, the mislabeled rate p_{mj} will arise. Although the correctness rate p_{dc} also arises, it is much slighter than p_{mj}. According to this figure, if the point spacings of the input data are the same, a reasonable configuration is obtained with d_r^1 and d_r^2 equal to the point spacing of the input data. Otherwise, we can set the parameters according to the variation tendency of the point density. For example, if the variation tendency of the point density is piecewise, we set d_r^1 smaller than the average point spacing in the edge detection step, and thus all the

edges can be ensured to be detected, next, we utilize the feature line tracing step to trace all the edges, then filter the traced segments with small number of points.

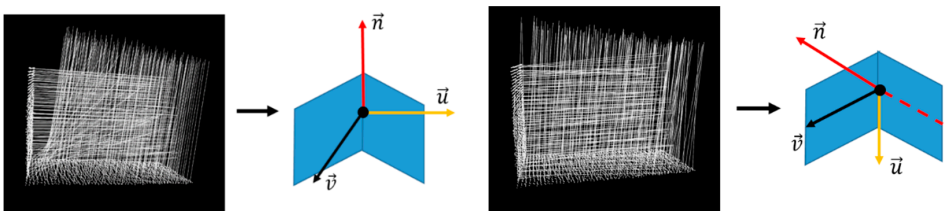


(**a**) (**b**)

Figure 9. Average values of correctness and mislabeled rates for different values of d_r^1 and d_r^2 : (**a**) results of the edge detection step with different values of ; and (**b**) results of the feature line tracing step with different values of d_r^2.

3.4. Normal Estimation

To demonstrate the feasibility of our RANSAC-Normal, we compare it with the PCA-Normal. The comparison is shown in Figure 10. When a point is a fold edge, there are two intersecting surfaces orthogonal to each other in its neighborhood. The PCA-Normal (see Figure 10a) is orthogonal to the tangent plane rather than one of the intersecting planes, while the RANSAC-Normal (see Figure 10b) is orthogonal to one of the intersecting planes. If we construct a coordinate frame based on the computed RANSAC-Normal, the other two axes $\vec{v}$ and $\vec{u}$ of the coordinate frame are located in one of the intersecting planes. Therefore, the RANSAC-Normal reaches the requirement of our method (see Section 2.2.4).



(**a**) (**b**)

Figure 10. Normal estimation of the neighborhood with two intersecting planes: (**a**) PCA-Normal $\vec{n}$; (**b**) RANSAC-Normal $\vec{n}$ estimated by our method.

3.5. Influence of Point Density

The point density of the input data mainly determines the details of a complex object, such as a chimney, step, or window. To analyze the influence of the input density on the results of the edge detection step and the feature line tracing step, we down-sampled a point cloud data of a window (see Figure 11a) with different point densities.

As shown in Figure 11c, the horizontal axis denotes the number of points in the down-sampled data. The light blue bar represents the ratio of the contained feature lines in the original data of a window to all feature lines in the window (denoted by r_1). When the data is down-sampled into different point densities, the value of r_1 decreases with the decreasing of point density. We can find that the ratio of feature lines in the detected edges to all feature lines (denoted by r_2) and the ratio of traced feature lines to all feature lines (denoted by r_3) are close to the ratio r_1. For example, when the

data is down-sampled to 6673 points, only the sixty percent of the original feature lines are contained, the values of r_1, r_2 and r_3 are close to 0.6 simultaneously, and both the correctness rates of the edge detection step and the feature line tracing step are close to 1.0. This indicates that as long as a feature line is contained in the data, our method can detect the edges and trace the feature line segment. Therefore, the experimental results demonstrate that the correctness rates of the edge detection step and the feature line tracing step are not influenced by the point density.

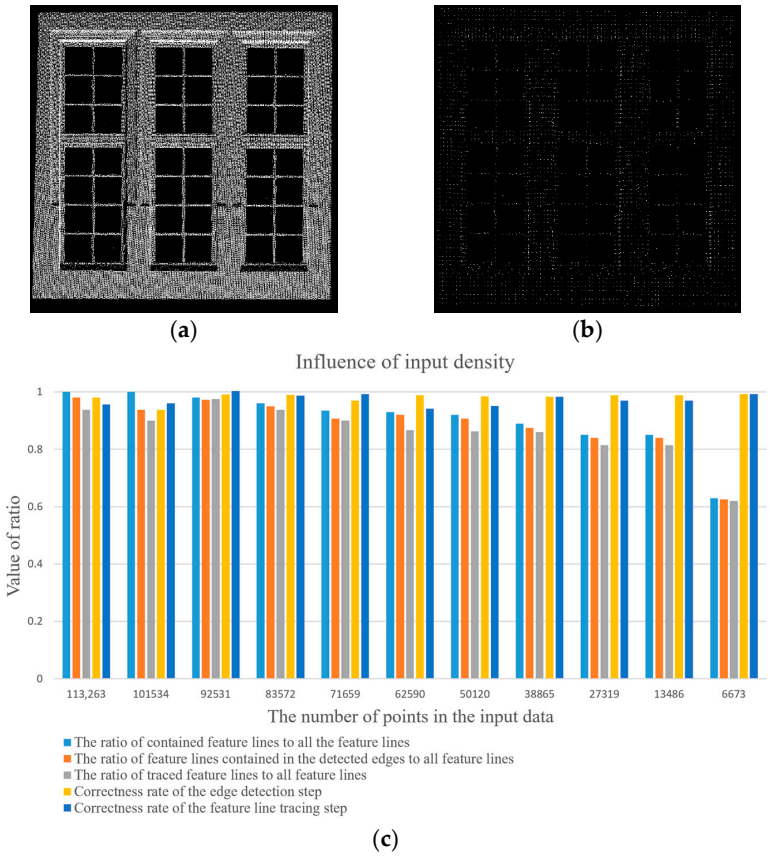


Figure 11. (**a**) Original input data without down-sampling; (**b**) the input data down-sampled to 6673 points; (**c**) correctness rates under different densities.

3.6. Results

With the discussed values of the parameters used in the proposed AGPN, the overall performance is evaluated on the aforementioned two testing sites.

To test the capability of the proposed AGPN detecting the defined types of edges presented in Section 1.1, a small scene in Site 2 is selected. The selected small scene and its details are shown in Figure 12a. Four surfaces exist in this scene, i.e., one curve surface rendered by red color and three planar surfaces rendered by blue, green, and yellow colors respectively. There are fold edges belonging to the intersection curves [49] and lines [13], and boundary elements belonging to the plane outlines. Three parts of the edges are marked by white rectangles. Specifically, the fold edges belonging to an intersection curve are the intersetions of the red curve surface and the green planar surface. The results of our proposed AGPN are shown in Figure 12b,c. Figure 12b shows the detected edges. We can find that all the defined edges are detected in the scene of interest. Figure 12c shows the traced feature line segments rendered by different colors. Specifically, the purple and orange segments correspond to the two intersection curves of the scene respectively, and other segments correspond to the intersection lines and outlines.

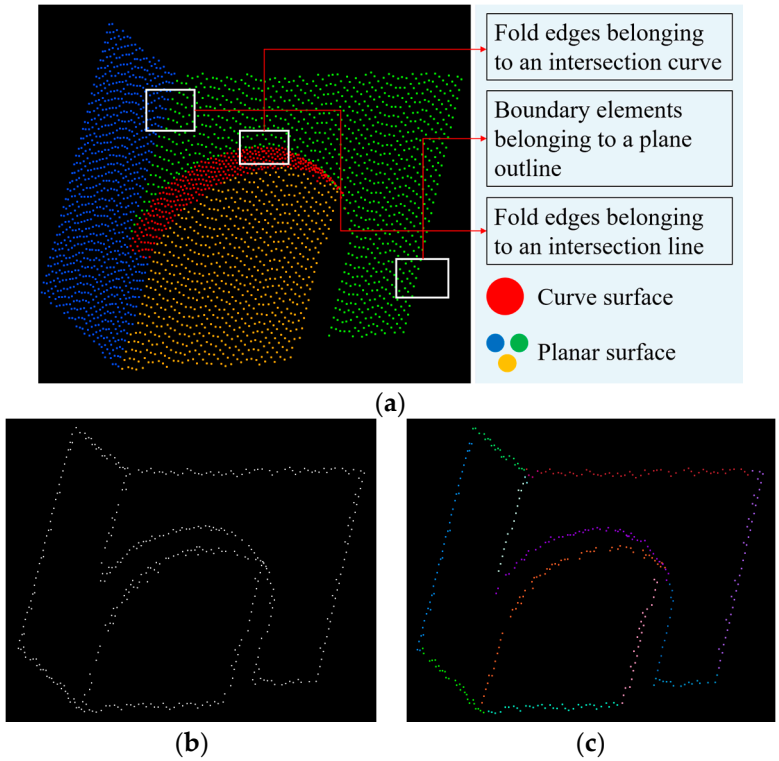


Figure 12. The results of a small area, (**a**) shows the four surfaces, and three kinds of edges in this area; (**b**) shows the edges detected by our method; (**c**) shows the feature line segments traced by our method.

In Site 1, the variation tendency of the point density is consecutive with the variation of the distances to the scanner. Therefore, we set the thresholds of d_r^1 and d_r^2 to the average point spacing 0.01. In Site 2, the variation tendency of the point density is piecewise, i.e., the point spacing in window areas (0.01) is much larger than that in non-window areas (0.005). According to the discussion in Section 3.3, we set the thresholds of d_r^1 and d_r^2 to 0.005 (the average point spacing in non-window areas). The parameters and their values used in the proposed algorithms are listed in Table 1. To clearly demonstrate the performance of the proposed method, the 3D edge detection results are overlaid on the original point cloud.

The results of Sites 1 and 2 are shown in Figures 13 and 14, and the accuracy evaluation results are listed in Table 2. In Table 2, the correctness rates p_{dc} and p_{dct} of Site 2 are higher than that of Site 1. The reason is that the variation range in Site 1 is larger than that in Site 2. According to the analysis in Section 3.3, when the values of d_r^1. and d_r^2 are unequal to the point spacing, the correctness rates will decrease. We can also find that the mislabeled rate p_{mj} of Site 1 is lower than that of Site 2. A close-up visual inspection shows that there are many windows in the second site data, the misjudgments mainly arise in the areas of windows. The reason is that the point spacings in the areas of windows are much larger than the ones in non-window areas. Moreover, the difference of point spacings between windows and non-window areas in Site 2 is larger than that in Site 1. According to the analysis in Section 3.3, when the parameter of distance threshold is smaller than the point spacing, the misjudgments will arise. Furthermore, the tendencies of the correctness rate p_{dc} and the mislabeled rate p_{mj} (in Figure 9a) validates that the mislabeled rates of the edge detection step in Table 2 is reasonable.

In Figures 13d–f and 14d–f, all the defined types of 3D edges are detected by our proposed method, and some details in the results of Sites 1 and 2 are presented. We can find that most of the edges in the window areas or walls with textures belong to intersection curves or intersection lines. Specially, most of the intersected planes or curve surfaces are narrow, which are difficult to be extracted by segmentation or clustering procedures for 3D-point clouds.

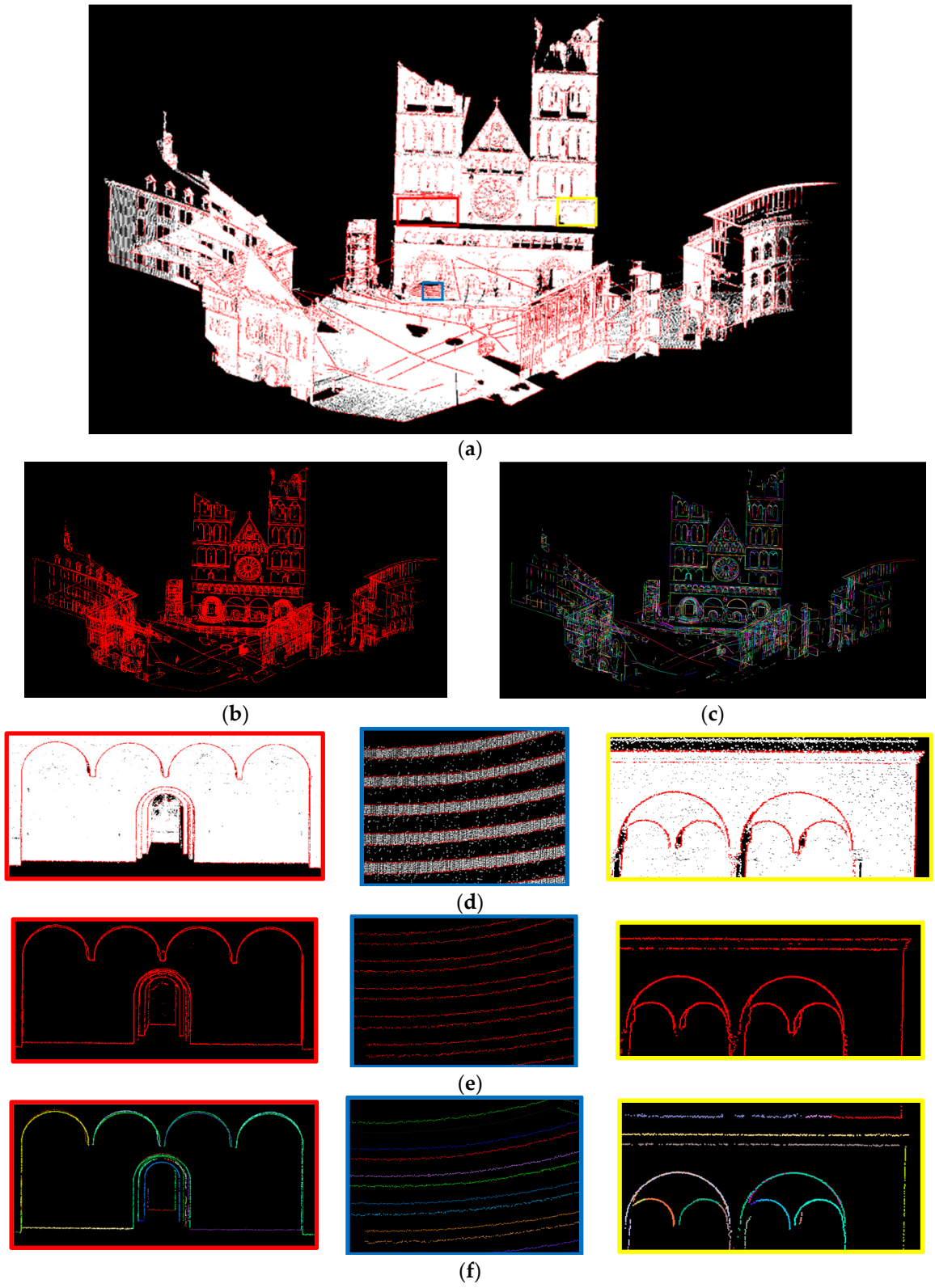


Figure 13. Results of Site 1: (**a**) edge detection result overlaid on the original input data; (**b**) edges; (**c**) traced feature line segments depicted in different colors; (**d**–**f**) details of edges and traced feature line segments demarcated by different colored outlines corresponding to (**a**).

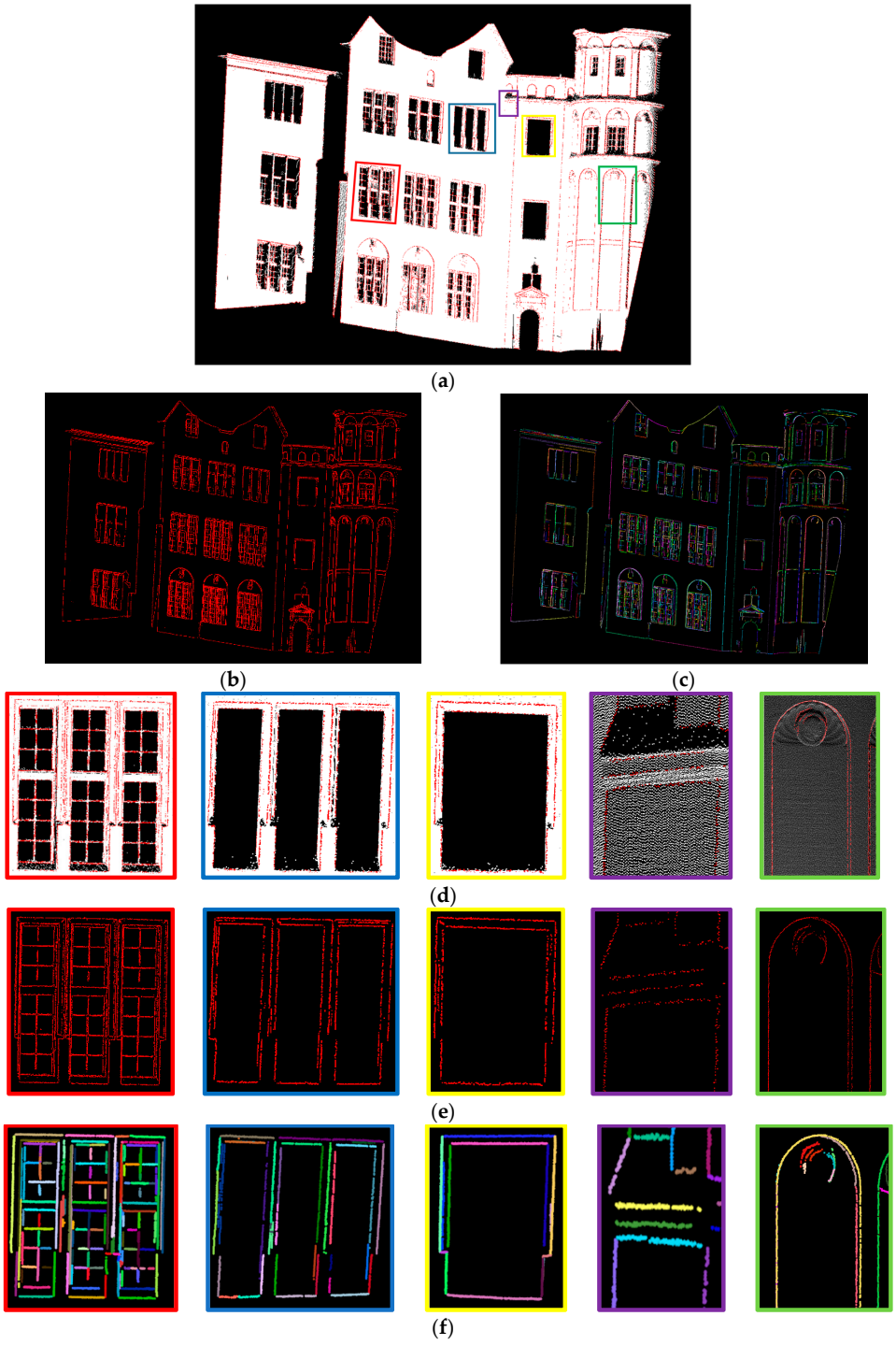


Figure 14. Results of Site 2: (**a**) edge detection result overlaid on the original input data; (**b**) edges; (**c**) traced feature line segments depicted in different colors; (**d**–**f**) details of edges and traced feature line segments, demarcated by different colored outlines corresponding to (**a**).

Table 2. Accuracy results of the proposed algorithms.

	Edge Detection		**Feature Line Tracing**	
	p_{dc} (%)	p_{mj} (%)	p_{dct} (%)	p_{mjt} (%)
Site 1	95.6	3.4	86.7	10.1
Site 2	98.1	5.2	87.9	5.3

3.7. Comparative Studies

In our comparison, we do not include all types of the existing methods. This paper compares the proposed AGPN method to "direct methods" because AGPN processes 3D-point clouds directly. In the proposed AGPN, there are two steps, i.e., edge detection step and feature line tracing step. We compare the existing methods with the two steps individually.

A boundary estimation method is presented in PCL, and it uses an angle criterion which is similar to the literature [41]. Comparative results of the boundary estimation method and our edge detection method are shown in Figure 15. Three subsets are selected from the testing datasets. The comparative results show that the proposed edge detection method can detect all types of edges irrespective of how complex the details of the objects are. However, the boundary estimation method in PCL is incapable of detecting fold edges. Table 3 lists the comparative accuracies of our edge detection method and the boundary estimation method.

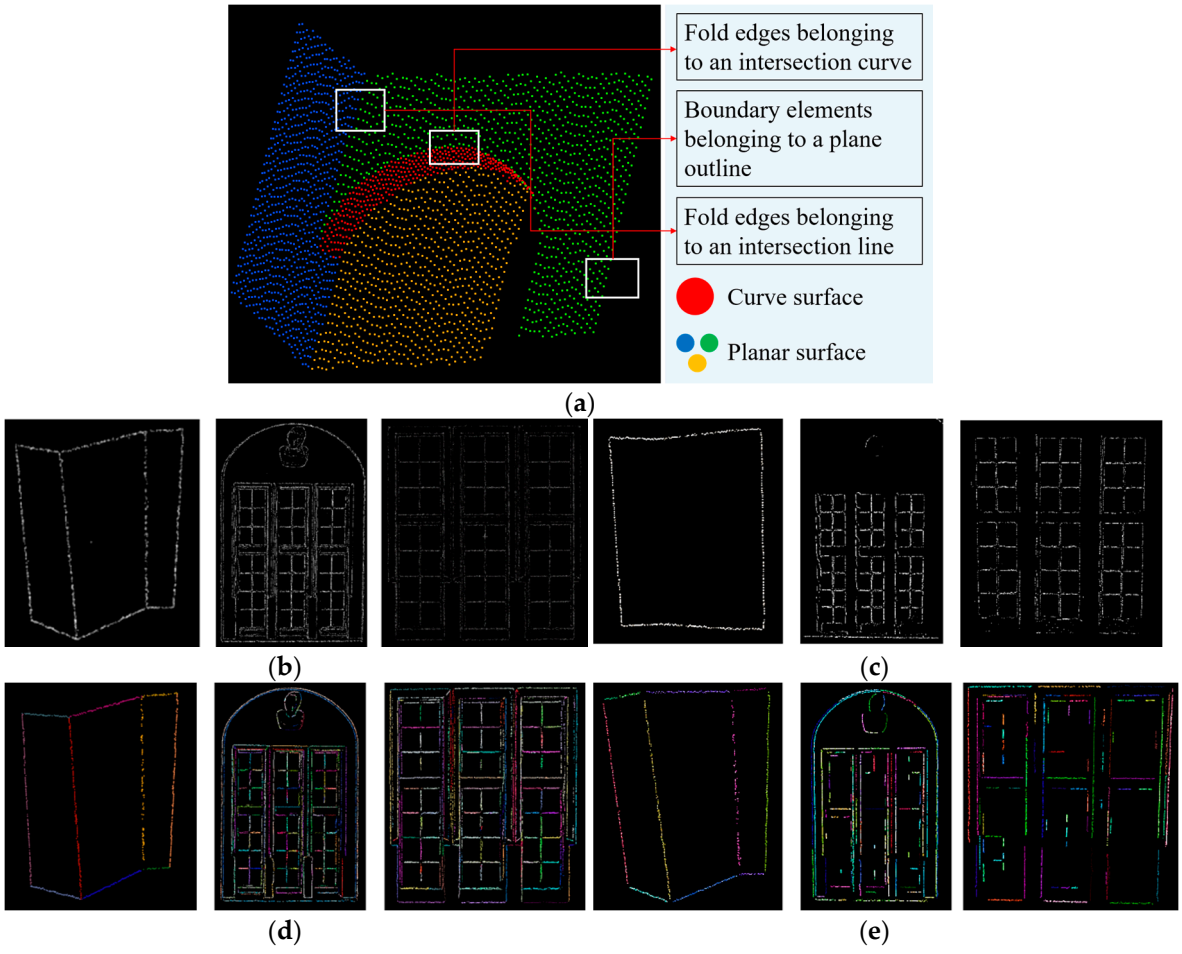


Figure 15. Comparison of the results of the AGPN and existing methods: (**a**) original input data; (**b**) edges detected by our edge detection method; (**c**) edges detected by PCL, where the results are optimal, obtained by training ten sets of parameters; (**d**) feature line segments traced by our method; (**e**) feature line segments traced by the method in the literature [13].

The reason is that the boundary estimation method detects 3D edges using only angle criterion, which is insufficient for detecting fold edges from unorganized point clouds. Besides, the boundary estimation method detects surface boundaries using the PCA-Normal, which ignores the influence of outliers and noises. Therefore, the boundary estimation method is incapable of dealing with an edge with a complex neighborhood.

Table 3. Comparative accuracy results.

	Our edge Detection Method		PCL Method		Our feature Line Tracing Method		Edge Points Clustered by [13]	
	p_{dc} (%)	p_{mj} (%)	p_{dc} (%)	p_{mj} (%)	p_{dct} (%)	p_{mjt} (%)	p_{dct} (%)	p_{mjt} (%)
Data1	100.0	0.0	80.0	0.0	100.0	0.0	100.0	0.0
Data2	88.3	6.3	52.6	2.7	90.3	19.6	50.6	3.4
Data3	86.8	5.2	60.3	5.5	88.9	14.1	31.0	4.4

An edge point clustering method has been proposed by the literature [13]. Comparative results of our feature line tracing method and the algorithm in the literature [13] are shown in Figure 15, where different traced feature line segments are rendered in different colors. Comparative accuracy results of our method and the method in the literature [13] are listed in Table 3. It can be seen that our feature line tracing method performs very well, and the tracing quality of it is obviously better than that of the algorithm in the literature [13].

The reason is that the literature [13] determines the principal direction of each edge point by analyzing a self-correlation matrix constructed by nearest neighbors, but ignoring outliers among the nearest neighbors. When several feature lines intersect, the neighborhood will not be a linear structure, resulting in the edges in this intersecting area will be missed.

4. Conclusions

In this paper, we propose an automated and effective method named AGPN, which detects 3D edges and traces feature lines from 3D-point clouds. There are two main steps in our proposed AGPN: edge detection and feature line tracing. In the first step, AGPN detects edges from 3D-point clouds. This step first analyzes the geometric properties of each query point's neighborhood, and then, combines RANSAC algorithm and angular gap criterion to label the query point as edge or non-edge. In the second step, AGPN traces feature lines from the detected 3D edges. This step is a hybrid method based on region growing and model fitting. In this step, we refine the neighborhood of each query point and redefine two growing criteria to overcome the uncertainties of the region growing procedure.

The contributions of the proposed AGPN method include: (1) image processing or point cloud preprocessing such as segmentation and object recognition are not needed, thereby reducing the complexity of the 3D-point cloud process; (2) the proposed edge detection method in AGPN can detect all the defined types of 3D edges and be insensitive to noise; (3) the feature line tracing step in AGPN is used to distinguish spatially-adjacent, collinear/coaxial lines in complex neighborhoods.

The experimental results show that our proposed AGPN can detect all the defined types of edges irrespective of how complex the details of the objects are. The feature line tracing step can trace feature lines in a complex neighborhood with several intersected or parallel lines. In comparison with state-of-the-art methods, the proposed AGPN can obtain superior results qualitatively and quantitatively. Moreover, we analyze the uncertainties of the results of our proposed method from two aspects, i.e., parameters tuning and influence of point density. In the analysis of the parameters tuning, we present the variation tendency of the correctness rate and the mislabeled rate with different parameters. Then, we achieve a good result according to the point spacing of the input data and the variation tendency. In the analysis of the influence of input density, the experimental results demonstrate that the two steps in our method are not influenced by the input density, though the details of a complex object mainly depend on it.

However, a limitation exists in that over-segmentation arises when the parameters of the feature line tracing procedure are strict or there is a gap in a long feature line. In the future work, we will attempt to solve this problem. In addition, it would be interesting to apply the proposed algorithms to object recognition in large-scale 3D-point clouds.

Acknowledgments: This research was funded by: (1) the General Program sponsored by the National Natural Science Foundations of China (NSFC) under Grant 41371405; (2) the Foundation for Remote Sensing Young Talents by the National Remote Sensing Center of China; and (3) the Basic Research Fund of the Chinese Academy of Surveying and Mapping under Grant 777161103.

Author Contributions: All of the authors contributed extensively to the work presented in this paper. Huan Ni proposed the method, implemented all the algorithms in the method and wrote this manuscript. Xiangguo Lin discussed the structure of the manuscript, named the method with Huan Ni, revised the manuscript, and supplied computers for experiments. Xiaogang Ning revised the manuscript. Jixian Zhang improved the manuscript, supported this study and protected the research process.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

There are five parameters in our proposed AGPN. Specifically, the first two parameters are the number of nearest neighbors K^1 and the distance threshold d_r^1 for the RANSAC plane model estimation in the edge detection step. The last three parameters are smooth direction threshold *sm_thr*, the number of nearest neighbors K^2 for *kd*-tree, distance threshold d_r^2 for the RANSAC line model estimation in the feature line tracing step. The following pseudo codes are the details.

AGPN

Input: Point cloud = $\{P\}$, parameter K^1, d_r^1, d_r^2, K^2, *sm_thr*.
Edge points $\{E\} \leftarrow \varnothing$
Feature line segments $\{FS\} \leftarrow \varnothing$
Edge detection step $\{E\} \leftarrow \Xi\left(K^1, d_r^1\right)$
Feature line tracing step $\{FS\} \leftarrow \Theta\left(E,\ K^2, d_r^2,\ sm\_thr\right)$
Return *Feature line segments* $\{FS\}$

Appendix B

The edge detection step in AGPN uses two parameters. Specifically, the first is the number of nearest neighbors K^1. The second is the distance threshold d_r^1 for the RANSAC model. Furthermore, there is a default threshold $\frac{\pi}{2}$ for the angular gap (G_θ). The following pseudo codes are the details.

3D Edge Detection

Input: Point cloud = $\{P\}$, parameters K^1, d_r^1.
Edge points $\{E\} \leftarrow \varnothing$
For $i = 0$ *to* size ($\{P\}$) **do**
  Current neighbors $\{N_c\} \leftarrow \varnothing$
  Find nearest neighbors of current point $\{N_c\} \leftarrow \varphi\left(P, i, K^1\right)$
  Current normal vector $n_c \leftarrow \{0,0,0\}$
  Current inlier list $\{I_c\} \leftarrow \varnothing$
  Compute current inlier list $\{I_c\}$ *using RANSAC* $\{I_c\} \leftarrow \Omega\left(N_c,\ d_r^1\right)$
  Normal optimization $n_c \leftarrow \phi\left(I_c\right)$
  If $P_i \notin I_c$ || size (I_c) <3 **then**
    Continue
  End If
  The first axis $u \leftarrow \{0,0,0\}$, *the second axis* $v \leftarrow \{0,0,0\}$
  Construct coordinate frame $u, v \leftarrow \Gamma\left(n_c\right)$
  Compute angular gap $G_\theta \leftarrow \Lambda\left(i, I_c, u, v\right)$
  If $G_\theta >= \frac{\pi}{2}$ **then**
    $\{E\} \leftarrow \{E\} \cup i$
  End If
End For
Return *Edge points* $\{E\}$

References

1. Ando, S. Image field categorization and edge/corner detection from gradient covariance. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *2*, 179–190. [CrossRef]
2. Frei, W.; Chen, C.C. Fast boundary detection: A generalization and a new algorithm. *IEEE Trans. Comput.* **1977**, *10*, 988–998. [CrossRef]
3. Shanmugam, K.S.; Dickey, F.M.; Green, J.A. An optimal frequency domain filter for edge detection in digital pictures. *IEEE Trans. Pattern Anal. Mach. Intell.* **1979**, *1*, 37–49. [CrossRef] [PubMed]
4. McIlhagga, W. The Canny edge detector revisited. *Int. J. Comput Vision* **2011**, *91*, 251–261. [CrossRef]
5. Meer, P.; Georgescu, B. Edge detection with embedded confidence. *IEEE Trans. Pattern Anal. Mach. Intell.* **2001**, *12*, 1351–1365. [CrossRef]
6. Lin, Y.B.; Wang, C.; Cheng, J. Line segment extraction for large scale unorganized point clouds. *ISPRS J. Photogramm. Remote Sens.* **2015**, *102*, 172–183. [CrossRef]
7. Guan, H.Y.; Li, J.; Cao, S.; Yu, Y. Use of mobile LiDAR in road information inventory: A review. *Int. J. Image Data Fusion.* **2016**, *3*, 219–242. [CrossRef]
8. Zhang, J.X.; Lin, X.G. Advances in fusion of optical imagery and LiDAR point cloud applied to photogrammetry and remote sensing. *Int. J. Image Data Fusion.* **2016**. [CrossRef]
9. Weinmann, M.; Jutzi, B.; Hinz, S.; Mallet, C. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS J. Photogramm. Remote Sens.* **2015**, *105*, 286–304. [CrossRef]
10. Rosenfeld, A.; Thurston, M. Edge and curve detection for visual scene analysis. *IEEE Trans. Computers.* **1971**, *20*, 562–569. [CrossRef]
11. Vanderheijden, F. Edge and line feature-extraction based on covariance-models. *IEEE Trans. Pattern Anal. Mach. Intell.* **1995**, *17*, 16–33. [CrossRef]
12. Vosselman, G.; Dijkman, S. 3D building model reconstruction from point clouds and ground plans. *Int. Arch. Photogramm. Remote Sens.* **2001**, *34*, 22–24.
13. Borges, P.; Zlot, R.; Bosse, M.; Nuske, S.; Tews, A. Vision-based localization using an edge map extracted from 3D laser range data. In Proceedings of the International Conference on Robotics and Automation (ICRA), Anchorage, AK, USA, 3–7 May 2010.
14. Demarsin, K.; Vanderstraeten, D.; Volodine, T.; Roose, D. Detection of closed sharp edges in point clouds using normal estimation and graph theory. *Comput. Aided Des.* **2007**, *39*, 276–283. [CrossRef]
15. Sampath, A.; Shan, J. Segmentation and reconstruction of polyhedral building roofs from aerial lidar point clouds. *IEEE Geosci. Remote Sens.* **2010**, *48*, 1554–1568. [CrossRef]
16. Sampath, A.; Shan, J. Clustering based planar roof extraction from LiDAR data. In Proceedings of the American Society for Photogrammetry Remote Sensing Annual Conference, Reno, NV, USA, 1–5 May 2006.
17. Pu, S.; Vosselman, G. Knowledge based reconstruction of building models from terrestrial laser scanning data. *ISPRS J. Photogramm. Remote Sens.* **2009**, *64*, 575–584. [CrossRef]
18. Overby, J.; Bodum, L.; Kjems, E.; Iisoe, P.M. Automatic 3D building reconstruction from airborne laser scanning and cadastral data using Hough transform. *Int. Arch. Photogramm. Remote Sens.* **2004**, *35*, 296–301.
19. Pu, S.; Vosselman, G. Extracting windows from terrestrial laser scanning. In Proceedings of the ISPRS Workshop Laser Scanning and Silvi Laser 2007, Espoo, Finland, 12–14 September 2007; pp. 320–325.
20. Boulaassal, H.; Landes, T.; Grussenmeyer, P. Automatic extraction of planar clusters and their contours on building facades recorded by terrestrial laser scanner. *Int. J. Archit Comput.* **2009**, *7*, 1–20. [CrossRef]
21. Sampath, A.; Shan, J. Building boundary tracing and regularization from airborne Lidar point clouds. *Photogramm. Eng. Remote Sens.* **2007**, *7*, 805–812. [CrossRef]
22. Wang, J.; Shan, J. Segmentation of LiDAR point clouds for building extraction. In Proceedings of the American Society for Photogrammetry Remote Sensing Annual Conference, Baltimore, MD, USA, 9–13 March 2009.
23. Nizar, A.; Filin, S.; Doytsher, Y. Reconstruction of buildings from airborne laser scanning data. In Proceedings of the American Society for Photogrammetry Remote Sensing Annual Conference, Reno, NV, USA, 1–6 May 2006.
24. Peternell, M.; Steiner, T. Reconstruction of piecewise planar objects from point clouds. *Comput. Aided Des.* **2004**, *36*, 333–342. [CrossRef]
25. Lafarge, F.; Mallet, C. Creating large-scale city models from 3D-point clouds: A robust approach with hybrid representation. *Int. J. Comput. Vision* **2012**, *1*, 69–85. [CrossRef]

26. Awrangjeb, M. Using point cloud data to identify, trace, and regularize the outlines of buildings. *Int. J. Remote. Sens.* **2016**, *3*, 551–579. [CrossRef]
27. Poullis, C. A framework for automatic modeling from point cloud data. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *11*, 2563–2575. [CrossRef] [PubMed]
28. Heo, J.; Jeong, S.; Park, H.K.; Jung, J.; Han, S.; Hong, S.; Sohn, H.-G. Productive high-complexity 3D city modeling with point clouds collected from terrestrial LiDAR. *Comput. Environ. Urban.* **2013**, *41*, 26–38. [CrossRef]
29. Rottensteiner, F.; Briese, C. Automatic generation of building models from LiDAR data and the integration of aerial images. In Proceedings of the International Society for Photogrammetry and Remote Sensing, Dresden, Germany, 8–10 October 2003; Volume 34, pp. 174–180.
30. Alharthy, A.; Bethel, J. Heuristic filtering and 3d feature extraction from LIDAR data. In Proceedings of the ISPRS Commission III, Graz, Austria, 9–13 September 2002.
31. Alharthy, A.; Bethel, J. Detailed building reconstruction from airborne laser data using a moving surface method. *Int. Arch. Photogramm. Remote Sens.* **2004**, *35*, 213–218.
32. Forlani, G.; Nardinocchi, C.; Scaiono, M.; Zingaretti, P. Complete classification of raw LIDAR and 3D reconstruction of buildings. *Pattern Anal. Appl.* **2006**, *8*, 357–374. [CrossRef]
33. Brenner, C. Building reconstruction from images and laser scanning. *Int. J. Appl. Earth Obs. Geoinf.* **2005**, *6*, 187–198. [CrossRef]
34. Li, H.; Zhong, C.; Hu, X.G. New methodologies for precise building boundary extraction from LiDAR data and high resolution image. *Sensor Rev.* **2013**, *2*, 157–165. [CrossRef]
35. Li, Y.; Wu, H.; An, R. An improved building boundary extraction algorithm based on fusion of optical imagery and LIDAR data. *Optik* **2013**, *124*, 5357–5362. [CrossRef]
36. Hildebrandt, K.; Polthier, K.; Wardetzky, M. Smooth feature lines on surface meshes. In Proceedings of the 3rd Eurographics Symposium on Geometry Processing, Vienna, Austria, 4–6 July 2005.
37. Altantsetseg, E.; Muraki, Y.; Matsuyama, K.; Konno, K. Feature line extraction from unorganized noisy point clouds using truncated Fourier series. *Visual Comput.* **2013**, *29*, 617–626. [CrossRef]
38. Huang, J.; Menq, C. Automatic data segmentation for geometric feature extraction from unorganized 3-D coordinate points. *IEEE Trans. Robot. Autom.* **2001**, *17*, 268–279. [CrossRef]
39. Gumhold, S.; Wang, X.; Macleod, R. Feature extraction from point clouds. In Proceedings of the 10th International Meshing Roundtable, Sandia National Laboratory, Newport Beach, CA, USA, 7–10 Octerber 2001.
40. Linsen, L.; Prautzsch, H. Local versus global triangulations. In Proceedings of the EUROGRAPHICS 2001, Manchester, UK, 2–3 September 2001.
41. Truong-Hong, L.; Laefer, D.F.; Hinks, T. Combining an angle criterion with voxelization and the flying voxel method in reconstructing building models from LiDAR data. *Comput. Aided Civ. Inf.* **2013**, *28*, 112–129. [CrossRef]
42. Linsen, L.; Prautzsch, H. Fan clouds—An alternative to meshes. In Proceedings of the 11th International Workshop on Theoretical Foundations of Computer Vision, Dagstuhl Castle, Germany, 7–12 April 2002.
43. PCL-The Point Cloud Library. 2012. Available online: http://pointclouds.org/ (accessed on 10 March 2016).
44. Bendels, G.H.; Schnabel, R.; Klein, R. Detecting holes in point set surfaces. *J. WSCG* **2006**, *14*, 89–96.
45. Zhang, J.X.; Lin, X.G.; Ning, X.G. SVM-based classification of segmented airborne LiDAR point clouds in urban area. *Remote Sens.* **2013**, *5*, 3749–3775. [CrossRef]
46. Vosselman, G.; Gorte, B.G.H.; Sithole, G.; Tabbani, T. Recognize structure in laser scanning point clouds. *Int. Arch. Photogramm. Remote Sens.* **2004**, *46*, 33–38.
47. CloudCompare-3D Point Cloud and Mesh Processing Software. 2013. Available online: http://www.danielgm.net/cc/ (accessed on 10 March 2016).
48. DTK-The 3D ToolKit. 2011. Available online: http://slam6d.sourceforge.net (accessed on 10 March 2016).
49. Wikipedia, the Free Encyclopedia. Available online: https://en.wikipedia.org/wiki/Intersection_curve (accessed on 5 May 2016).