remote sensing

MDPI

Article

Incremental and Enhanced Scanline-Based Segmentation Method for Surface Reconstruction of Sparse LiDAR Data

Weimin Wang [1,*], Ken Sakurada [1] and Nobuo Kawaguchi [1,2]

[1] Graduate School of Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan; sakurada@nagoya-u.jp (K.S.); kawaguti@nagoya-u.jp (N.K.)

[2] Institutes of Innovation for Future Society, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan

* Correspondence: weimin@ucl.nuee.nagoya-u.ac.jp; Tel.: +81-452-789-4544

Academic Editors: Gonzalo Pajares Martinsanz and Prasad S. Thenkabail
Received: 26 September 2016; Accepted: 9 November 2016; Published: 22 November 2016

Abstract: The segmentation of point clouds is an important aspect of automated processing tasks such as semantic extraction. However, the sparsity and non-uniformity of the point clouds gathered by the popular 3D mobile LiDAR devices pose many challenges for existing segmentation methods. To improve the segmentation results of point clouds from mobile LiDAR devices, we propose an optimized segmentation method based on Scanline Continuity Constraint (SLCC) in this work. Unlike conventional scanline-based segmentation methods, SLCC clusters scanlines using the continuity constraints in terms of the distance as well as the direction of two consecutive points. In addition, scanline clusters are agglomerated not only into primitive geometrical shapes but also irregular shapes. Another downside to existing segmentation methods is that they are not capable of incremental processing. This causes unnecessary memory and time consumption for applications that require frame-wise segmentation or when new point clouds are added. In order to address this, we propose an incremental scheme—the Incremental Recursive Segmentation (IRIS), that can be easily applied to any segmentation method. IRIS is achieved by combining the segments of newly added point clouds and the previously segmented results. Furthermore, as an example application, we construct a processing pipeline consisting of plane fitting and surface reconstruction using the segmentation results. Finally, we evaluate the proposed methods on three datasets acquired from a handheld Velodyne HDL-32E LiDAR device. The experimental results verify the efficiency of IRIS for any segmentation method and the advantages of SLCC for processing mobile LiDAR data.

Keywords: point clouds; LiDAR; surface reconstruction; incremental processing; automatic 3D modeling; 3D segmentation; sparsity; planar simplification

1. Introduction

3D mobile LiDAR devices are becoming rapidly popular due to their long effective range, wide Field of View (FOV) and high accuracy. They are widely used in many fields, such as civil engineering [1,2], autonomous driving and robotics. In these applications, automated processing of the point cloud, such as semantic extraction, object recognition and 3D reconstruction is usually performed to improve the efficiency of reconstruction and the modeling or accuracy of recognition. Consequently, segmentation—the first step in most of these processes—is a very important and indispensable operation.

There exist many segmentation algorithms, such as Region Growing [3] and RANSAC [4]. However, two main challenges are encountered when these are applied to LiDAR data. The first issue

arises because of the sparsity and non-uniformity of the point cloud acquired by mobile LiDAR devices when compared with the point cloud generated by depth sensors or static LiDAR devices. Sparsity and non-uniformity greatly increase the rate of over-segmentation and miss-segmentation. The second issue is associated with the ability to process the point cloud incrementally. The segmentation time increases drastically as the scale of the point cloud increases. Furthermore, the time-consuming computation has to be performed again if a new point cloud is added. This can be avoided if incremental segmentation is available.

In order to address the first challenge, we propose a Scanline Continuity Constraint (SLCC) segmentation method that is optimized for segmenting point clouds from 3D mobile LiDAR devices. SLCC takes full advantage of the characteristics of scanlines. Each scanline is divided into several clusters according to the *continuity constraints* imposed due to the distance and direction change between two consecutive points. The clustered scanlines are then agglomerated into 3D segments of arbitrary shapes. To address the second challenge, we propose a solution inspired by the frame-wise processing of point clouds in autonomous driving and robotics applications. We propose the Incremental RecursIve Segmentation (IRIS) method, which makes use of the previous segmentation results. In this, a newly added set of point clouds is segmented first and then the new segments are combined directly with the segments that are processed previously.

Three main contributions are described in this work. We propose a novel segmentation method that is able to overcome the sparsity and non-uniformity of a point cloud from mobile LiDAR devices. Our second contribution is an incremental scheme that can be easily adapted to any segmentation algorithm. An example using Region Growing with and without applying our proposed incremental scheme demonstrates the efficiency. Finally, as a segmentation application, we construct a processing pipeline that performs plane fitting and surface reconstruction segment by segment in parallel. The final processing results (Figure 1b,d) of point clouds (Figure 1a,c) acquired from a handheld Velodyne HDL-32E LiDAR sensor verify the feasibility of the proposed methods.

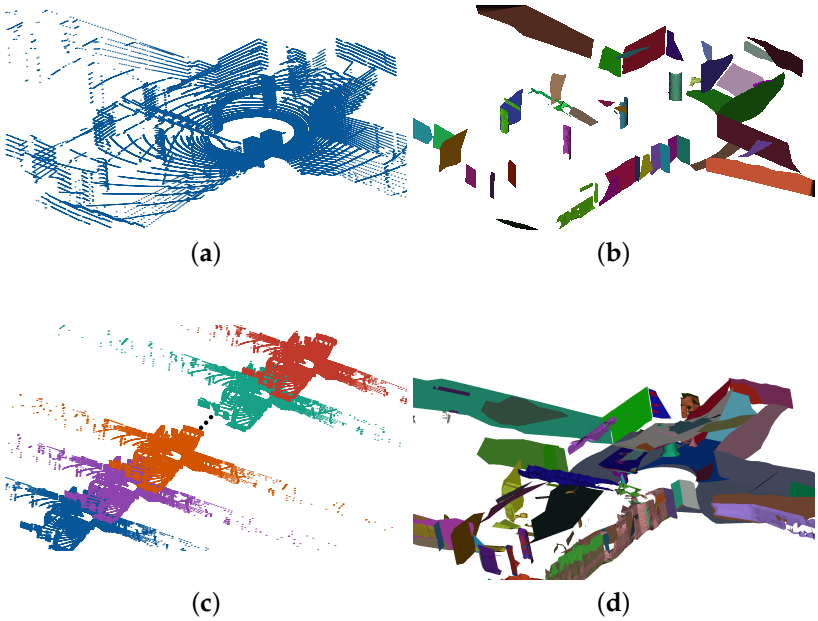


Figure 1. Automatic segmentation and modeling for a single frame of a point cloud or multiple frames. Left (**a**,**c**) show the single frame of spare point cloud input and multiple frames of sparse point cloud input respectively; Right (**b**,**d**) are output results; the surface is reconstructed using segmented information. Different segments are visualized using different colors.

The rest of this paper is structured as follows. Related work and open challenges are discussed in Section 2. An overview of the proposed approach, SLCC segmentation and IRIS to final surface

reconstruction, is provided in Section 3. The improved segmentation algorithm for mobile LiDAR data SLCC is presented in Section 4. The details of the incremental scheme IRIS for application in any segmentation algorithm are described in Section 5. The pipeline to implement automated refined surface reconstruction is explained in Section 6. The efficiency of IRIS and the advantages of SLCC for LiDAR data are evaluated and verified in Section 7. As an application of the proposed segmentation method, surface reconstruction results of mobile LiDAR data are also shown in Section 7. Finally, the conclusion and future work are presented in Section 8.

2. Related Work

With the increasing popularity of 3D scanning devices, there is a need for efficient semantic extraction and surface reconstruction from raw unorganized point clouds. We mainly focus on mobile LiDAR devices such as Velodyne HDL-32E utilized in this work, which usually generates sparse and non-uniform point clouds every frame, unlike the ones generated by depth sensors or static LiDAR devices. In this section, we only list research related to the segmentation and surface reconstruction of mobile LiDAR data.

The first step in the recognition process involves the segmentation of the point cloud. For a dense and uniformly distributed point cloud, Region Growing in [3,5] is a representative method. For every point in the point cloud, the algorithm estimates their curvature value and the normal vector of the approximate plane constructed by its kNN points. The recursion process starts from the point with the least curvature and the point cloud is segmented according to the intersection angle of the points' normal vectors. However, as sparsity and non-uniformity of a point cloud increases, it becomes harder to perform the segmentation [6]. Model fitting based on Random Sample Consensus (RANSAC) [7,8] is another popular method for segmentation in which the point cloud is fitted to primitive geometrical models, such as a plane, sphere, or cylinder. This approach performs well when detecting shapes from a point cloud that can be represented by several primitive shape models. However, it becomes inefficient as the number of primitive shapes increases. In this work, we apply RANSAC to detect whether a segment could be represented by a planar model.

Voxelization of a point cloud is also applied for segmentation [9]. Papon et al. [10] propose a method that segments the point cloud according to the connectivity of the voxels into a supervoxel, which is similar to a superpixel for image segmentation. This method could be applied for segmenting all shapes. Nevertheless, memory consumption is a serious problem for a large-scale point cloud. In contrast to depth sensors, LiDAR devices scan the environment and obtain the 3D points one-by-one. Based on the features of scanlines, many methods have been proposed for different purposes, such as people detection, segmentation and point cloud registration [11–15].

The usual way to reconstruct large-scale environments with mobile LiDAR data is by mapping several sparse point clouds acquired from different positions and angles into a single point cloud. Although there are many excellent methods to map them with minimal drift [12,16,17], measurement noise and registration errors are inevitable. These make the planar surface look foggy and thick. To regularize the point cloud, planar shapes are estimated using Region Growing or RANSAC and the points on the estimated planes are redirected using preset regularity relationships, such as parallel, orthogonal or coplanar [18,19]. These methods are mainly used to process dense and uniform point cloud.

Redundant points consume storage, memory and computation unnecessarily. Downsampling is often used to preprocess the point cloud data, and most of the points can be represented by primitive shapes. Especially, point clouds that can be represented by planar shapes, planar decimation [20], or polygon boundary extraction [1,21] permit drastic removal of redundant data. Although convex boundary extraction works for most cases, concave boundary represents the shapes better [22].

Triangulation-based surface reconstruction is a classic and intuitive method [23]. Delaunay triangulation [24] and its variant, the constrained Delaunay triangulation [25], are widely applied in

3D games or movies for surface reconstruction. To improve the efficiency and robustness to noise, quadtree-based triangulation has also been proposed [26,27] for planar surface reconstruction.

Automatic surface reconstruction and modeling also attracts interest due to utility in applications such as VR games, virtual interior design and so on. There have been some studies for automatic room detection and modeling from very dense point clouds [2,28]. In [2,28], wall boundaries are extracted from a dense point cloud acquired using static LiDAR and mapped onto a 2D floor plane to construct a cell complex. This enables the modeling of the room polyhedra. In this work, we aim to implement surface reconstruction with semantic segmentation for sparse and non-uniform point clouds.

3. Notations and System Overview

3.1. Notations

To explain the proposed method in detail, we use the point cloud obtained using a Velodyne HDL-32 LiDAR device as a sample and define the following notations (see Figure 2).

- A complete 360° sweep by the LiDAR is denoted as one frame $\mathcal{F}$.
- The point cloud acquired in the i-th frame $\mathcal{F}_i$ is denoted as $\mathcal{P}_i$.
- The scanline in $\mathcal{P}_i$ is denoted as $\mathcal{L}_{(i,j)}$. In the case of Velodyne HDL-32, scanlines are $\{\mathcal{L}_{(i,1)}, \mathcal{L}_{(i,2)}, \ldots, \mathcal{L}_{(i,32)}\}$.
- All the scanlines of $\mathcal{P}_i$ are divided into several clusters line by line, denoted by $\{\mathcal{C}_{(i,1)}, \mathcal{C}_{(i,2)}, \ldots, \mathcal{C}_{(i,k)}\}$.
- Clustered scanlines are then agglomerated into final segments $\{\mathcal{S}_{(i,1)}, \mathcal{S}_{(i,2)}, \ldots, \mathcal{S}_{(i,l)}\}$ of $\mathcal{P}_i$.

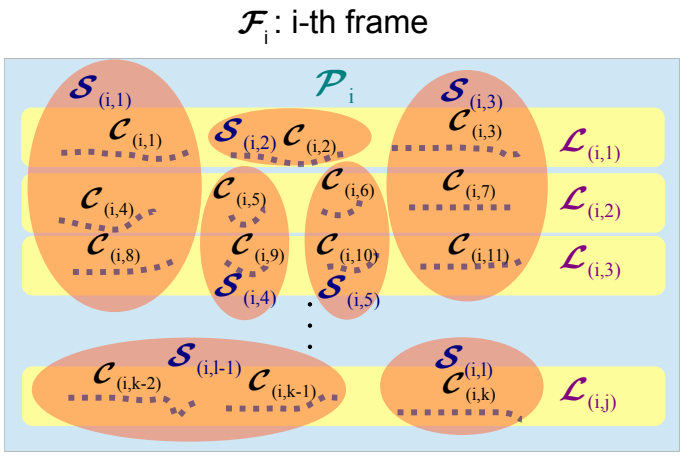


Figure 2. Notations defined in this paper.

3.2. System Overview

The pipeline of our proposed system is shown in Figure 3. When the environment is perceived by a LiDAR device, a series of continuous frames $\{\mathcal{P}_1, \mathcal{P}_2, \ldots\}$ are generated. The generated point clouds are processed using the following three steps.

- SLCC Segmentation

Frame-wise segmentation is often required for applications such as autonomous robots to recognize objects for route generation, manipulation and interaction with humans. In the case of 3D mobile LiDAR, we propose an enhanced scanline-based method to optimize the segmentation results for a sparse and non-uniform point cloud in a frame of mobile LiDAR devices. Details of clustering for scanlines and agglomerating scanline clusters are presented in Section 4.

- IRIS Segmentation

As the scale of point clouds increases gradually, the computation time and memory consumption of segmentation increase drastically. On the other hand, when point clouds of new areas or details are added, the newly registered point cloud has to be segmented again. This results in extra time and memory consumption. Hence, we consider increasing the efficiency of segmentation by introducing an incremental scheme for segmentation as explained in Section 5.

- Surface Reconstruction

To demonstrate the segmentation performance and present an application example, surfaces are reconstructed using segmentation information. Once the point cloud has been segmented, surface reconstruction is performed for each segment independently shown as the right part in Figure 3, to which parallel processing can be easily applied. In Section 6, reconstruction based on plane fitting and alpha shape is described in detail.

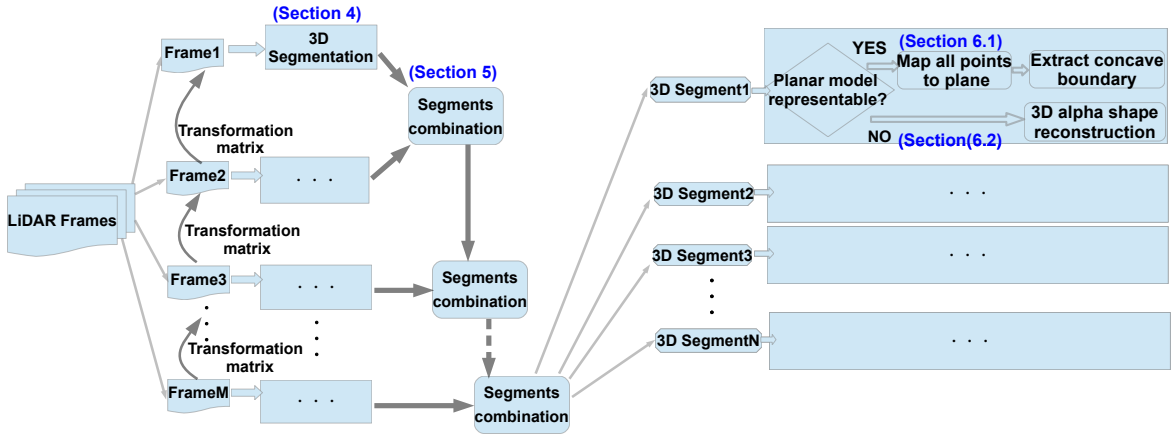


Figure 3. Block diagram of the pipeline for a processing point cloud.

4. Scanline Continuity Constraint (SLCC) Segmentation

In this section, we introduce the SLCC segmentation method for sparse and non-uniform point clouds. We explain the method of Scanline clustering and then agglomeration of these clusters.

4.1. Clustering of Scanlines

Omnidirectional LiDAR devices usually obtain 360° range information in a horizontal direction by spinning laser beams along the vertical axis. We cluster the scanlines based on the observations discussed next (illustrated in Figure 4). The distance between two consecutive points when observed from a particular object does not change drastically along the direction of scanning. The change of the angle between two vectors formed by two groups of consecutive points from an object maintains continuity. The second observation has special implications when segregating two adjacent but perpendicular parts, such as two intersecting walls of a corner.

Based on these two observations, we divide the scanlines from different objects into different segments. The pseudocode of the process is shown in Algorithm 1.

Algorithm 1: Scanline clustering

Input : The point cloud $\mathcal{P}_i$ generated by the i-th frame $\mathcal{F}_i$
c_{th}: coefficient for jumping's threshold ($c_{th} = 5$ in this work)
$\Delta\theta$: horizontal angular resolution of device ($\Delta\theta$ is 0.165° for Velodyne HDL-32 LiDAR)
θ_{th}: threshold for intersection angle of two vectors ($\theta_{th} = 80°$ in this work)

Output: The set of clustered Scanlines

```
Scanline cluster {c} ← ∅
list of Scanline clusters {C} ← ∅
divide P_i into {L_(i,1), L_(i,2), ..., L_(i,j)}
foreach L in L_(i,j) do
    foreach p_m in L do
        d_th ← CalcDisThre(p_m, Δθ, c_th)
        d ← CalcDistance (p_m)
        θ ← CalcAngle (p_m)
        if d < d_th and θ > θ_th then
            add point p_m to {c}
        else
            add {c} to {C}
            {c} ← ∅
            add point p_m to {c}
        end
    end
end
return {C}
```

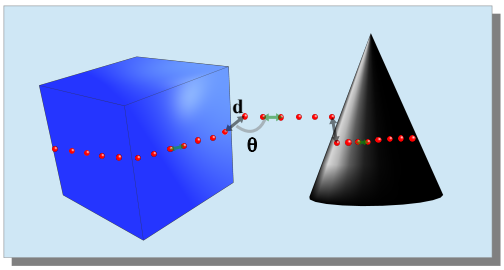


Figure 4. Two constraints of Scanline continuity. 1. d : distance of two consecutive points (green lines show the distance of two consecutive points from the same object, black lines show the same from different objects); 2. θ: the angle between two vectors of two groups of consecutive points. Two points are considered to be from two objects if either d or θ exceeds the user-defined thresholds.

First, we divide every frame of the point cloud $\mathcal{P}_i$ into scanlines $\{\mathcal{L}_{(i,1)} \ \ldots, \mathcal{L}_{(i,j)}\}$ where j equals the number of laser beams of the 3D LiDAR. For each scanline $\mathcal{L}$, we perform clustering by judging whether two consecutive points belong to a same object along the scanning direction. For a point $\boldsymbol{p}_m$ in $\mathcal{L}$, if the Euclidean distance (function $CalcDistance(\boldsymbol{p}_m)$ in Algorithm 1) between $\boldsymbol{p}_m$ and its next point $\boldsymbol{p}_{m+1}$ is less than the distance threshold d and the intersection angle (function $CalcAngle(\boldsymbol{p}_m)$ in Algorithm 1) between the vectors $(\boldsymbol{p}_{m-1} - \boldsymbol{p}_m)$ and $(\boldsymbol{p}_{m+1} - \boldsymbol{p}_{m+2})$ is less than the angular threshold θ_{th}, the points $\boldsymbol{p}_m$ and $\boldsymbol{p}_{m+1}$ are considered to have been scanned from the same object and are clustered into the same scanline segment. In order to make the algorithm more robust, more consecutive points can be selected to estimate the vectors.

With respect to the distance threshold parameter, we notice that the theoretical interval between two consecutive points changes as the angular resolution or measurement range changes (Figure 5).

Hence, we introduce an adaptive distance threshold (function $CalcDisThre(\boldsymbol{p}_m, \Delta\theta, c_{th})$ in Algorithm 1) by multiplying the theoretical interval with a user-defined constant c_{th}. Furthermore, while calculating the theoretical interval, the effect of the incident angle between the laser and the scanned object is also considered (See Figure 6). The final d_th is approximately calculated as:

$$d_{th} \approx \frac{r \cdot \Delta\theta \cdot c_{th}}{\sin\alpha} \tag{1}$$

where the incident angle α is estimated to be the first principal vector when the first principal component ratio of $\{\boldsymbol{p}_{m-n}, \ldots, \boldsymbol{p}_m\}$ is greater than 0.9 after the Principal Component Analysis (PCA). Otherwise it is set to 90°.

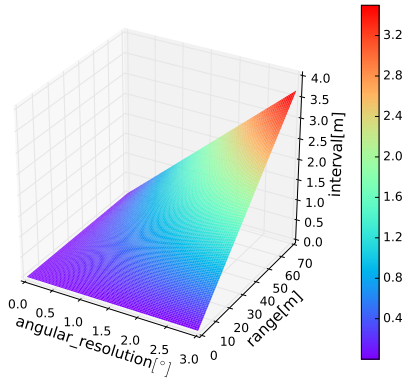


Figure 5. Theoretical interval of two consecutive points for different angular resolution and ranges. We can see that the distance of consecutive points increases greatly as range increases. This is the reason why we introduce an adaptive threshold for clustering scanlines.

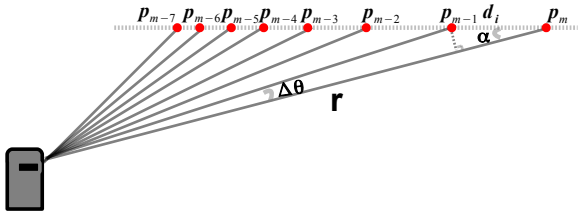


Figure 6. The effect of the incident angle to the theoretical interval of point $\boldsymbol{p}_m$. $\Delta\theta$ is the resolution angle in a horizontal direction. r is the range from point $\boldsymbol{p}_m$ to the LiDAR. α is the incident angle. $\boldsymbol{d_i}$ represents the theoretical interval.

There are mainly two differences between the proposed method and the method of scanline clustering proposed in [14]. The first difference is that the threshold for the Jump Distance Cluster (JDC) is empirically decided as a fixed value in [14], while we also consider the fact that the distance between two consecutive points increases as the scanning range increases. To take this into account, we introduce the adaptive threshold which is especially significant for a large-scale scene. The second difference is that we consider not only continuity in distance, but also in the directional sense. This ensures that scanlines from adjacent but irrelevant objects will be divided separately, decreasing the possibility of under-segmentation.

4.2. Agglomeration of Scanline Clusters

Scanline clusters are processed individually to train an AdaBoost classifier for people detection in [14]. In our work, we agglomerate these clusters for 3D segmentation using a method that can be adapted for arbitrary 3D object classification and recognition.

Two criteria are used to decide whether two scanline clusters belong to a single segment. The first one is the Euclidean distance between the centroids of the two clusters. The centroid point $\boldsymbol{p}_c$ of a cluster $\mathcal{C}$ is calculated using Equation (2).

$$\boldsymbol{p}_c = \frac{\sum_{i=1}^{n}(\boldsymbol{p}_i)}{n} \tag{2}$$

where $\boldsymbol{p}_i = (x_i, y_i, z_i) \in \mathbb{R}^3$ and n is the number of points in the scanline cluster $\mathcal{C}$. As in the case of scanline clustering, we apply an adaptive threshold to judge the distance between two centroids with the only difference being that the angular resolution is in the vertical direction.

The other criterion is the similarity of two clusters. This is decided using the PCA results of the two clusters. By decomposing all the 3D points in one scanline cluster on the three principal basis with PCA, we compute the components' ratio vector $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)^T$ on the basis vectors $\boldsymbol{M} = (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\mu}_3)^T$, where λ_1 to λ_3 represent the first to the third component ratio on the base $\boldsymbol{\mu}_1$ to $\boldsymbol{\mu}_3$ respectively.

Any two clusters $\mathcal{C}_j$ and $\mathcal{C}_k$ are considered to be from the same object if their PCA results satisfy the following conditions.

$$\begin{cases} |\boldsymbol{\lambda}^j - \boldsymbol{\lambda}^k| & < s_r \\ \sum_{i=1}^{3} \frac{\lambda_i^j + \lambda_i^k}{2} \cdot \frac{<\boldsymbol{\mu}_i^j, \boldsymbol{\mu}_i^k>}{|\boldsymbol{\mu}_i^j| \cdot |\boldsymbol{\mu}_i^k|} & > s_b \end{cases} \tag{3}$$

Here, s_r and s_b represent user-defined thresholds related to the similarity of two clusters on the basis of the component ratio vectors and the basis vectors respectively. Algorithm 2 shows the process of agglomeration of the scanline clusters in detail.

Algorithm 2: Agglomeration of Scanline clusters

Input : $\{\mathcal{C}_{(i,k)}\}$: Set of clustered Scanlines of $\mathcal{P}_i$
c_{th_g}: coefficient of distance's threshold for agglomeration ($c_{th\_g} = 5$ in this work)
s_r: ratios' threshold for similarity ($s_a = 0.2$ in this work)
s_b: basis vectors' threshold for similarity ($s_b = 0.9$ in this work)

Output: The set of 3D segments for $\mathcal{P}_i$

list of segments $\{\mathcal{S}\} \leftarrow \emptyset$
while $\{\mathcal{C}_{(i,k)}\}$ *is not empty* **do**
  segment $\{s\} \leftarrow \emptyset$
  take one cluster out from $\{\mathcal{C}_{(i,k)}\}$ and add it into $\{s\}$
  foreach *Scanline cluster* $\mathcal{C}_a$ **in** $\{\mathcal{C}_{(i,k)}\}$ **do**
    $\boldsymbol{p}_{c\_a} \leftarrow$ `CalcCentroid`$(\mathcal{C}_a)$
    foreach *scan lin cluster* $\mathcal{C}_b$ *in* $\{s\}$ **do**
      $\boldsymbol{p}_{c\_b} \leftarrow$ `CalcCentroid` $(\mathcal{C}_b)$
      $d_{th\_g} \leftarrow c_{th\_g} * |\boldsymbol{p}_{c\_a}|$
      $d_g \leftarrow$ `CalcDistance`$(\boldsymbol{p}_{c\_a}, \boldsymbol{p}_{c\_b})$
      if $d_g < d_{th\_g}$ **and** `IsSimilar`$(\mathcal{C}_a, \mathcal{C}_b, s_r, s_b)$ **then**
        add $\mathcal{C}_b$ to segment
        remove $\mathcal{C}_b$ from $\{\mathcal{C}_{(i,k)}\}$
      end
    end
  end
  add $\{s\}$ to $\{\mathcal{S}\}$
end
return $\{\mathcal{S}\}$

5. Incremental Recursive Segmentation (IRIS)

By applying the processes described in previous sections, we can segment a point cloud $\mathcal{P}_a$ into $\{\mathcal{S}_{a,l}\}$. When a point cloud with new spatial information or details $\mathcal{P}_b$ is acquired, the segmentation of the combined point cloud of $\mathcal{P}_a$ and $\mathcal{P}_b$ is performed. The conventional approach used to achieve this consists of aligning and combining the two point clouds into a single new point cloud and performing segmentation on the new combined point cloud. As stated previously, this procedure is not efficient in terms of time and memory consumption, especially for applications that require the point cloud to be frame-wise segmented or updated frequently. We propose an incremental scheme to address this drawback. As the first step, the transformation matrix from $\mathcal{P}_b$ to $\mathcal{P}_a$ needs to be estimated. There are many existing methods, such as ICP [29], Generalized-ICP [30], LOAM [12] and semantic alignment [31] to calculate the transformation matrix. To simplify the explanation, $\mathcal{P}_b$ represents the point cloud that has been transformed. In practical environments, point clouds of dynamic objects, such as pedestrians, can be removed to get a clear reconstruction either by tightening the thresholds of the proposed method or identifying them with the method in [32].

5.1. Combination of Segments from Different Point Clouds

$\mathcal{P}_b$ is segmented into $\{\mathcal{S}_{(b,l)}\}$ first. Next, we randomly *pick out* one segment $\mathcal{S}_a$ from $\{\mathcal{P}_{(a,j)}\}$ and check its relation to each segment of $\{\mathcal{P}_{(b,j)}\}$ successively. If $\mathcal{S}_a$ and some segment $\mathcal{S}_b$ from $\mathcal{P}_b$ are judged as being from the same object by the method in Section 5.3, we add $\mathcal{P}_b$ to $\mathcal{P}_a$. If there is no paired segment for $\mathcal{S}_a$ found after traversing all the segments in $\{\mathcal{S}_{(b,j)}\}$, $\mathcal{S}_a$ is considered to be a segment of the final output. The same process is repeated until $\{\mathcal{P}_{(a,j)}\}$ is empty. The final result consists of the unpaired segments of $\{\mathcal{S}_{(a,j)}\}$ and the expanded $\{\mathcal{S}_{(b,j)}\}$. Algorithm 3 shows the pseudocodes of this procedure.

5.2. Recursive Process for a More General Situation

Algorithm 3 only explains the case of two point clouds. This method can also be extended to recursively process more general situations. For a series of continuous point clouds $\{\mathcal{P}_1, \ldots, \mathcal{P}_j\}$, the last point cloud $\mathcal{P}$ is segmented and combined with segments of the combined point cloud of $\{\mathcal{P}_1, \ldots, \mathcal{P}_{(j-1)}\}$ once they have been segmented. Otherwise, we keep searching forward until the segmented point cloud is available. The function *IRIS* in Algorithm 4 represents the recursion of Algorithm 4.

5.3. Similarity of 3D Segments

To combine segments from different point clouds, quantitative criteria are needed to decide whether two segments belong to the same object (function $IsSameObject(\mathcal{S}_a, \mathcal{S}_b)$ in Algorithm 3). Examples of intuitive criteria are the range overlap of two segments or the distance between two segment centroids. However, in actual situations, it is very difficult to arrive at a decision based only on these conditions because of the sparsity and non-uniformity of point clouds and the presence of measurement and mis-registration errors. We thus introduce a linear classifier to solve this problem. Its principle is explained in the following.

An object can be scanned into different point clouds generated by different frames swept in different positions. After these point clouds are registered, the points that present the same object will have a high degree of overlap. We use the degree of overlap of points from the segments to judge whether the segments are from the same object. An example is illustrated in Figure 7. The grey pyramid on the left side represents a real object. It is scanned by three frames (denoted by Frame 1, 2, and 3) in different positions. Then, three point clouds generated by these frames are segmented with the proposed SLCC method independently. Three groups of points in the middle column show the segments that represent the pyramid in the three point clouds. After the registration of these point

clouds, three segments are mapped to the same coordinate system and shown as the right point cloud. As they fully overlap each other, we consider these segments as representing the same object.

Algorithm 3: Combine segments of two point clouds for IRIS

Input : $\{\mathcal{S}_{(a,l)}\}$: Segments of point cloud $\mathcal{P}_a$
$\mathcal{P}_b$: Newly acquired point cloud
Output: Segments of the combined point cloud

The set of segments $\{\mathcal{S}\} \leftarrow \emptyset$
Segment $\mathcal{P}_b$ into $\{\mathcal{S}_{(b,l)}\}$
while *$\{\mathcal{S}_{(a,l)}\}$ is not empty* **do**
  foreach $\mathcal{S}_a$ **in** *$\{\mathcal{S}_{(a,l)}\}$* **do**
    foreach $\mathcal{S}_b$ **in** *$\{\mathcal{S}_{(b,l)}\}$* **do**
      if *IsSameObject*$(\mathcal{S}_a, \mathcal{S}_b)$ **then**
        $\mathcal{S}_b \leftarrow$ `MergeTwoSegments` $(\mathcal{S}_a, \mathcal{S}_b)$
        remove $\mathcal{S}_a$ from $\{\mathcal{S}_{(a,l)}\}$
        break
      end
    end
    if *not* **break then**
      add $\mathcal{S}_a$ to $\{\mathcal{S}\}$
      remove $\mathcal{S}_a$ from $\{\mathcal{S}_{(a,l)}\}$
    end
  end
end
$\{\mathcal{S}\} \leftarrow \{\mathcal{S}\} \cup \{\mathcal{S}_{(b,j)}\}$
return *$\{\mathcal{S}\}$*

Algorithm 4: IRIS

Input : A series of continuous point clouds $\{\mathcal{P}_1, \ldots, \mathcal{P}_j\}$
Output: Segments of the combined point cloud from $\mathcal{P}_1$ to $\mathcal{P}_j$

if *the number of point clouds equals to 1* **then**
  return `SegmentPointCloud` *($\mathcal{P}_1$)*
else
  if *the combined point cloud of $\{\mathcal{P}_1, \ldots, \mathcal{P}_{j-1}\}$ is segmented* **then**
    return `CombineSegments` (`IRIS` *($\{\mathcal{P}_1, \ldots, \mathcal{P}_{j-1}\}$)*,`SegmentPointCloud` *($\mathcal{P}_j$)*)
  else
    return `IRIS` *($\{\mathcal{P}_1, \ldots, \mathcal{P}_{j-1}\}$)*
  end
end

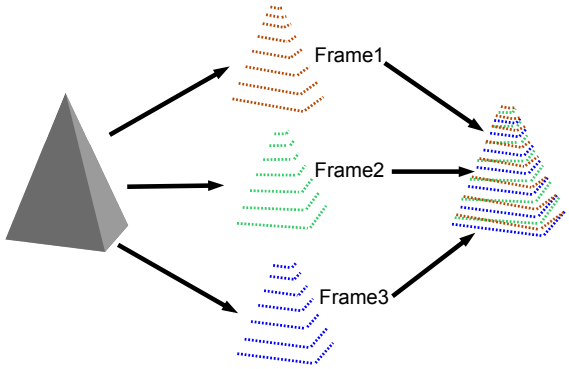
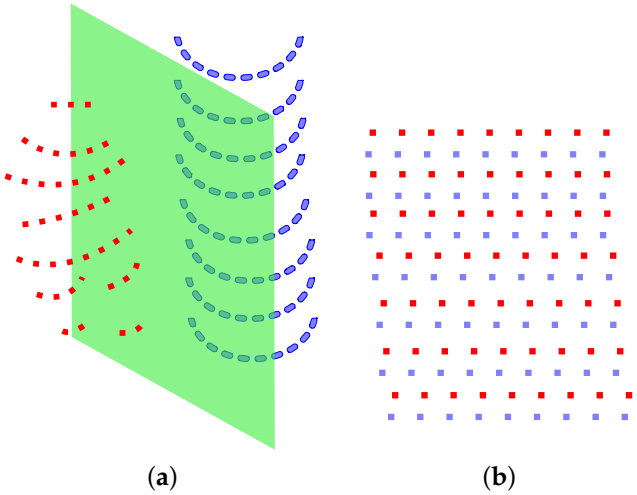


Figure 7. Example for combining segments. The middle red, green, blue point clouds are three scanned frames of the left object from different positions and angles, the right one shows the combined segment.

However, unlike polygons in 2D or polyhedrons in 3D, it is not easy to define the rate of overlap of 3D point clouds. We utilize the concept of the linear classifier to quantify this factor. If there exists a 3D plane that separates the two segments completely, we consider no overlap present (refer to Figure 8a). Otherwise, they are considered as being overlapped with each other (for example in Figure 8b). The degree of overlap is defined as $\frac{m_T+n_T}{m+n}$, where m, n are the numbers of $m+n$ points in two segments and m_T, n_T represent the number of points distinctly separated by the plane. The optimal 3D plane that maximizes the rate of overlay is estimated with a perceptron classifier. Coordinates of the 3D point are used as features. For two segments that consist of m and n points, respectively, m samples labeled as 0 and n samples labeled as 1 are input to train the perceptron. The training score indicates the degree of overlap. For example, if the score equals 1, it means there exists a 3D plane that completely separates the two segments with the rate of the overlay being 0. If the score of the training is less than the threshold defined by $\max\left(0.99, \frac{m}{m+n}, \frac{n}{m+n}\right)$ in this study, the two segments are considered to belong to the same object.



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

Figure 8. Linear classifier to judge whether two segments should be combined. (**a**) A separating plane exists between the red and blue point cloud, so we consider them as derived from different objects; (**b**) No plane could completely separate them, so we consider them as being from the same object.

In practical situations, a registration error of multiple sets of point clouds is inevitable. Figure 9 illustrates a situation affected by a registration error. In Figure 9, the red and blue points represent segments of the same planar object such as a wall, in two different point clouds. They are supposed to be mapped to the same plane as in Figure 8a and considered as being from the same object after

registering the point clouds. Nevertheless, a tiny "crevice" is generated due to such a registration error. As a result, there exists a plane that completely separates the two segments, which are judged as belonging to different objects. To ensure robustness to the registration error, we impose a prerequisite on the shapes of the two segments before sending them to the perceptron classifier. If two segments are fitted to two parallel planes, they are still considered to belong to the same object if the distance between the two planes is less than a certain tolerance (0.5 m in this study).

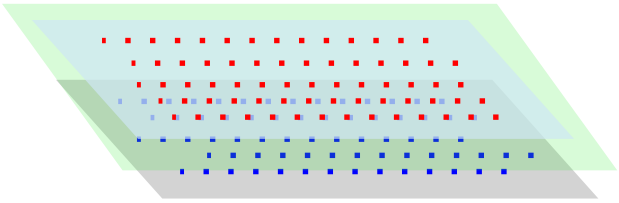


Figure 9. A challenge caused by registration error. The red and blue point clouds show scanned data of a plane such as a wall or roof, and they should be intersecting as in Figure 8b. However, a separating plane exists in the gap caused by registration errors, which indicates that they are from different objects. For this situation, we map the red and blue points into the same plane and then judge with a linear classifier.

6. Surface Reconstruction

Surface reconstruction makes the point cloud appear more real and plays an important role in applications such as VR games, 3D games, and 3D animation. The quality of surface reconstruction relates to the density of the point cloud, with more points increasing the memory and time consumption. The process of modeling results in the best reconstruction with the least points for surfaces that can be represented by primitive shapes.

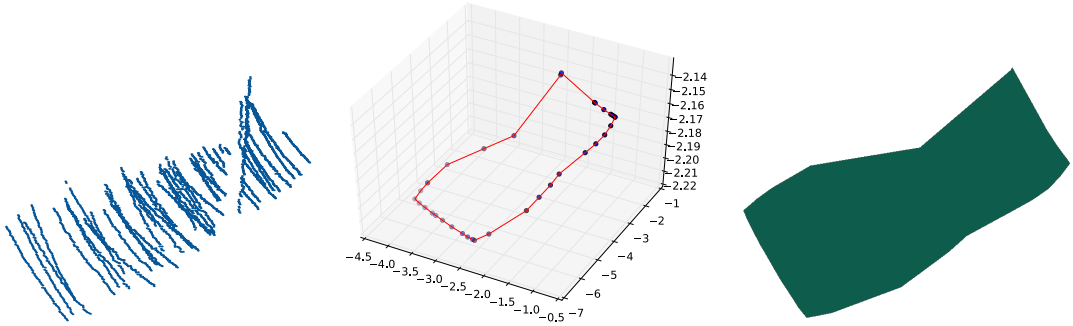
In this section, we describe the processing procedure after performing combination of segments from different frames. For each combined segment, we use RANSAC to detect whether a segment is a planar shape and estimate the parameters of the plane if it is. A segment is considered to be a planar if the percentage of inlier points exceeds a threshold. For segments that are not planar, the surface is reconstructed based on the 3D alpha shape.

6.1. Planar Fitting and Polygon Boundary Extraction

For each combined segment, we try to estimate whether it is a plane by using the RANSAC planar model. If the percentage of detected inlier points exceeds the threshold, we process this segment as a planar shape. We consider the outliers to be caused by measurement or registration errors. To preserve more geometrical details and eliminate the effects of registration errors, we first map all the points including outliers to the estimated plane. Then the original 3D segment is converted into a 2D segment and the boundary points are extracted by detecting its concave hull based on the alpha shape as shown in Figure 10.

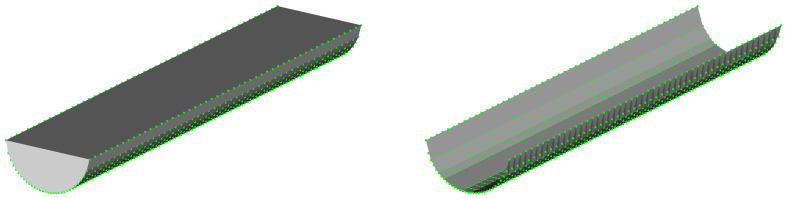
6.2. Surface Reconstruction for Non-Planar Shapes with Alpha Shape

If the percentage of detected inlier points is less than the threshold value, we consider the segment to be a non-planar shape. Because the points obtained by LiDAR devices are not uniformly distributed, it is difficult to generate the mesh with one point and its kNN points. Therefore, for these non-planar segments, we reconstruct the surface based on the 3D alpha shape [33,34] implemented by a Visualization Toolkit (VTK) [35]. An intuitive understanding of the alpha shape is that it uses circles or spheres with an alpha-value-related radius to find the hull of a set of unorganized 2D or 3D points. The alpha shape prevents unnecessary surface reconstruction as opposed to the Delaunay triangulation method, as show in Figure 11.



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

Figure 10. Processing for planar segments. (**a**) An input point cloud segment that is detected as a planar shape; (**b**) All points are mapped in the detected plane, and boundary points are extracted; (**c**) Planar surface is reconstructed.



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

Figure 11. Surface reconstruction comparison. Green points represents vertexes of the point cloud. (**a**) Shows the reconstructed surface by 3D Delaunay triangulation, where a 3D convex hull is constructed; (**b**) shows the result based on alpha shape. We can see that surfaces are better reconstructed based on alpha shape rather than Delaunay triangulation.

7. Results and Discussion

We evaluate our method with a series of datasets collected from a handheld Velodyne HDL-32E LiDAR sensor. It is possible to collect data from places that are narrow or have obstacles (such as stairs, places after disasters and in the crowd) using this method of data acquisition. However, such scenarios introduce more challenges in data processing. The characteristics of each dataset (listed in Table 1) and experimental results on each dataset are shown in this chapter. On account of space limitation, only representative segmentation results and surfaces results are shown and discussed in this section. All evaluations were performed on a workstation running Ubuntu 16.04 with an Intel Xeon E5-2609 CPU @ 2.40GHz and 16GB of RAM.

7.1. Datasets

Three datasets were collected for evaluating our proposed method. According to the area under consideration, they are labeled as Corridor, Lobby and Underground shopping mall, and represent small, medium and large-scale environments respectively. All the three datasets generate approximately 70,000 3D points per frame. A sample point cloud and panoramic image of each of the datasets is shown in Figure 12.

7.2. Time Performance of IRIS

We apply the Region Growing and IRIS Region Growing algorithms on all three datasets. Time consumption is tested from one frame of point cloud to sixteen frames of point clouds. The time of transforming the point cloud is not considered for both methods. For example, if n frames of point

clouds are processed, the time consumption in the case of Region Growing is the time taken to segment the combined point cloud of n frames with Region Growing. For IRIS Region Growing, it consists of the time taken to segment the n-th frame of the point cloud and the time taken to combine segments of the n-th frame and the previous $n-1$ frames. The results are shown in Figure 13. We can see that the processing time increases drastically for Region Growing as the number of frames increases, while it increases slowly for IRIS Region Growing, which effectively utilizes the previous segmentation results. It is interesting that for some frames, as the number of frames increases, the processing time decreases instead (for example, 11 frames in Figure 13c). That is because the number of segments in the last frame is relatively small, and consequently the combination time of segments decreases.

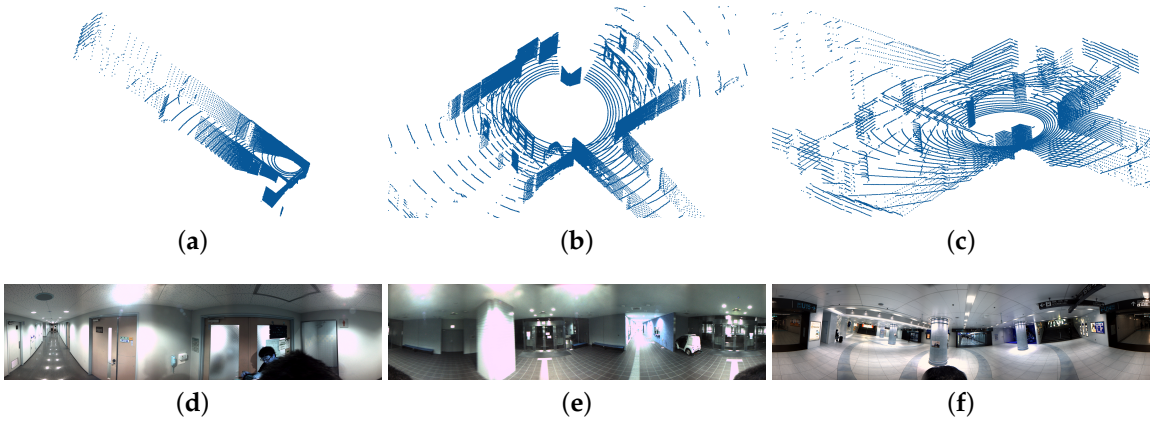


Figure 12. Three datasets are used in this work. (**a**–**c**) Point clouds of the Corridor, Lobby and Underground shopping mall dataset respectively; (**d**–**f**) Panoramic images of the three datasets.

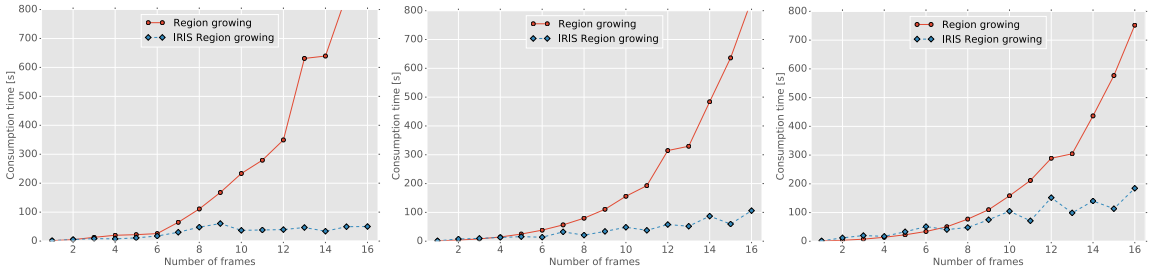


Figure 13. Time performance of IRIS . Both Region Growing and IRIS Region Growing are applied to the three datasets from one frame to sixteen frames of point clouds. (**a**–**c**) Results of the Corridor, Lobby and Underground shopping mall are shown respectively. The time consumption of IRIS Region Growing is calculated without considering the computation time of previous frames.

Table 1. Statistics of the evaluated datasets

Dataset	Corriodr	Lobby	Underground Shopping Mall
Area (m^3)	13.7 × 16.4 × 3.2	73.8 × 60.5 × 6.0	137.8 × 37.4 × 16.1
mean (m)	1.9	6.5	7.8
std. (m)	1.8	4.5	4.8
Points per frame		70,000	
Number of frames	24	12	16

7.3. Segmentation Performance

7.3.1. Scanlines Clustering

We compare the result of clustering scanlines for a single frame using the method in [14] and our proposed method. The results of clustering are shown in Figure 14. The left part in Figure 14b will cause under-segmentation after the agglomeration of scanline clusters, while the left part in Figure 14d will not, by virtue of having taken the continuity in direction into consideration. The right part in Figure 14b will cause under-segmentation after the agglomeration of scanline clusters, while the right part in Figure 14d will not. This is because of using an adaptive threshold and considering the incident angle. In general, our proposed method clusters scanlines more appropriately and suppresses over- and under-segmentation of the point cloud.

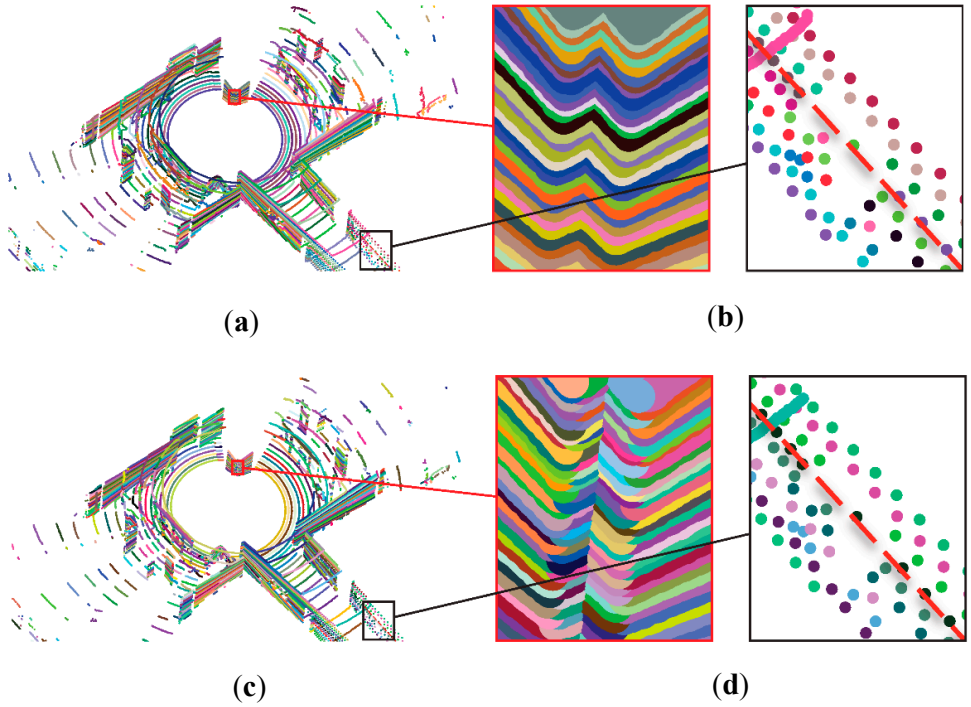


Figure 14. Scanline clustering comparison. (**a**,**b**) show the result of the method in [14], and (**c**,**d**) show the result of our proposed method for Scanline clustering. Magnified images are shown in (**b**,**d**). Compared with the left part in (**d**), the left part in (**b**) will cause under-segmentation after the agglomeration of Scanline clusters. Compared with the right part in (**d**), the right part in (**b**) will cause over-segmentation after the agglomeration of scanline clusters (see the red dash line).

7.3.2. Segmentation

In the next step, the scanlines clustered in the last step are agglomerated into 3D segments. We compare the segmentation results of the Region Growing algorithm and our proposed SLCC for a frame of the point cloud (see Figure 15). Some example results of irregularly shaped objects segmented by SLCC are shown in Figure 16. For multiple frames of point cloud, the results of Region Growing, IRIS Region Growing and IRIS SLCC are compared in Figure 17.

From Figure 15, we can see that our proposed SLCC segmentation method performs better than the Region Growing method in terms of over-segmentation (bound by blue lines in Figure 15b) and miss-segmentation (the magnified images in Figure 15d). For the case of multiple frames of point clouds, on comparing Figure 17a,c, we observe that Region Growing is prone to over-segmentation mainly due to the non-uniformity of the point cloud. The incremental scheme IRIS is able to suppress the over-segmentation besides increasing the efficiency as shown in Section 7.2. This is because the process

(Algorithm 3) of combining segments compresses the previous segments ($\{\mathcal{S}_{(a,l)}\}$ in Algorithm 3) into new segments ($\{\mathcal{S}_{(b,l)}\}$ in Algorithm 3). Thus, the number of final segments is comparable to the number of new segments. However, over-segmentation (upper rows in Figure 17b,d,f) increases for all three methods, as the number of frames increases.

As we stated in Section 4, the SLCC segmentation method agglomerates the scanline clusters according to their relative similarity. Compared with conventional scanline based methods, SLCC is able to segment not only objects of primitive geometrical shapes but also irregularly shaped objects. The thresholds in Algorithms 1 and 2 can be adjusted accordingly to adapt for the desired segmentation in different environments. For example, for the constraint of direction, the continuity in direction can be relaxed to segment pedestrians in the scanlines clustering step.

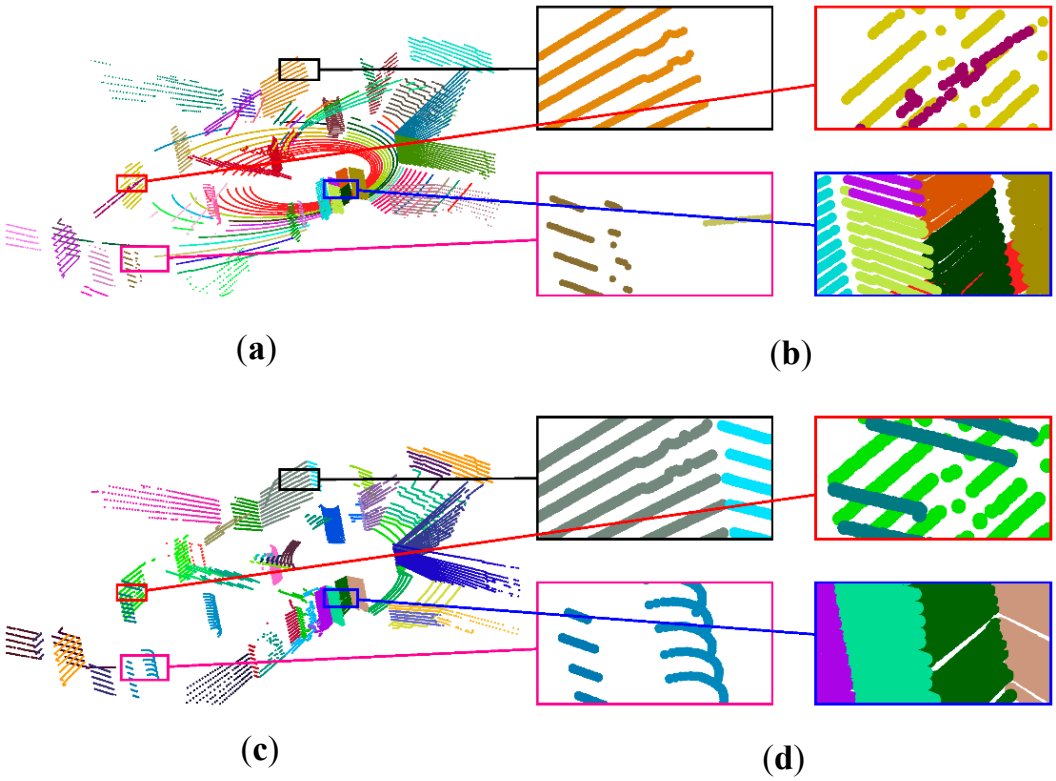


Figure 15. Segmentation results for one frame of point cloud. (**a**,**b**) show results with Region Growing, and (**c**,**d**) show results with our proposed SLCC. From the magnified images in the right column, our proposed SLCC performs better in avoiding miss-segmentation (marked by boxes with black, red and magenta lines) and over-segmentation (marked by the box with blue lines).

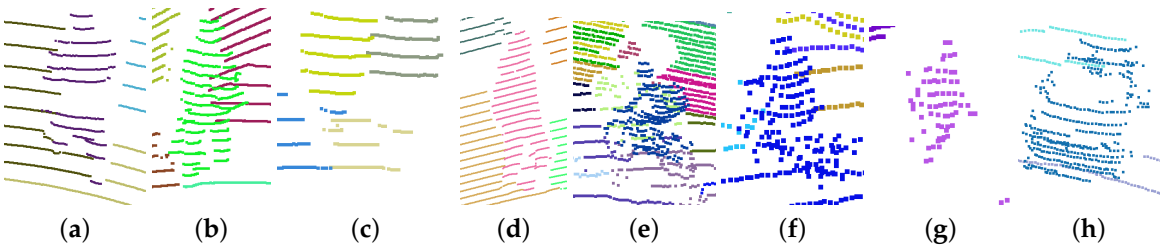


Figure 16. Examples of irregularly shaped objects segmented by SLCC. (**a**–**d**) Pedestrian, pedestrian, two pedestrians side by side, pedestrian segmented from the Underground shopping mall dataset; (**e**–**h**) cyclist, cyclist, pedestrian, car segmented from the KITTI dataset [36].

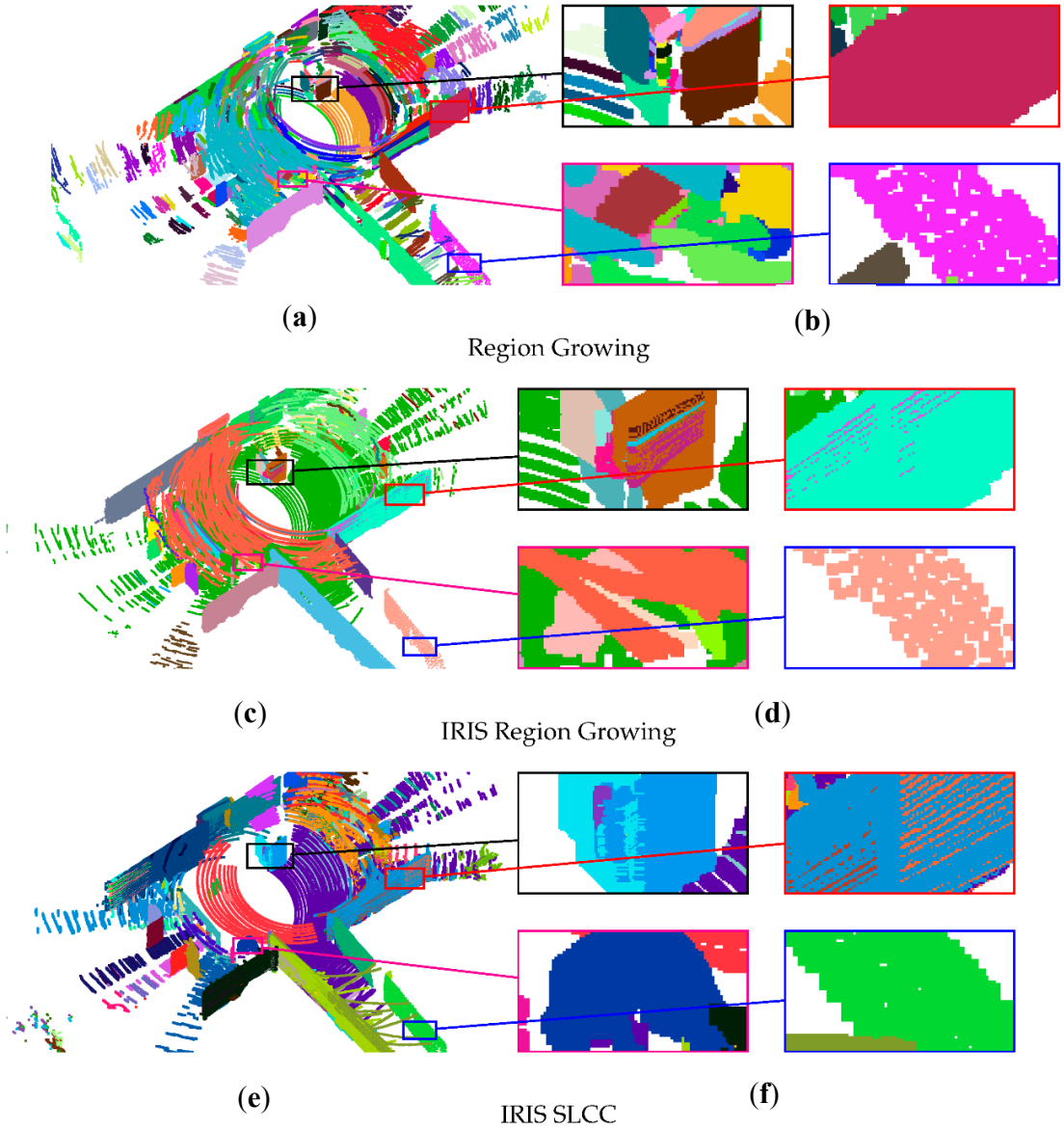


Figure 17. Comparison of segmentation results on multiple frames. Results of Region Growing, IRIS Region Growing and SLCC are shown from upper to lower rows respectively. Magnified images are shown in the right column. (**a**,**b**) Region Growing is prone to over-segmentation due to the sparsity and non-uniformity; (**c**,**d**) Over-segmentation of Region Growing is also suppressed by IRIS; (**e**,**f**) A small car is well segmented by IRIS SLCC (marked by the box with magenta lines).

7.4. Surface Reconstruction

We fit every individual segment to the plane model for segments that are planar and reconstruct the surface based on the alpha shape for segments that are not planar. The results are visualized using the open source software Visualization Toolkit (VTK) [35].

Surface reconstruction of the frame point cloud is shown in Figure 18. We observe, from Figure 18b, that there are many creases when the surface is constructed directly without surface fitting. On implementing the process discussed in Section 6.1, the creases are flatter, as shown in Figure 18d. Surface reconstruction of the segments of multiple frames is shown in Figure 19. Multiple colors on the planar surface of magnified images are examples of over-segmentation due to registration error. More results of surface reconstruction can be found in Figure 20.

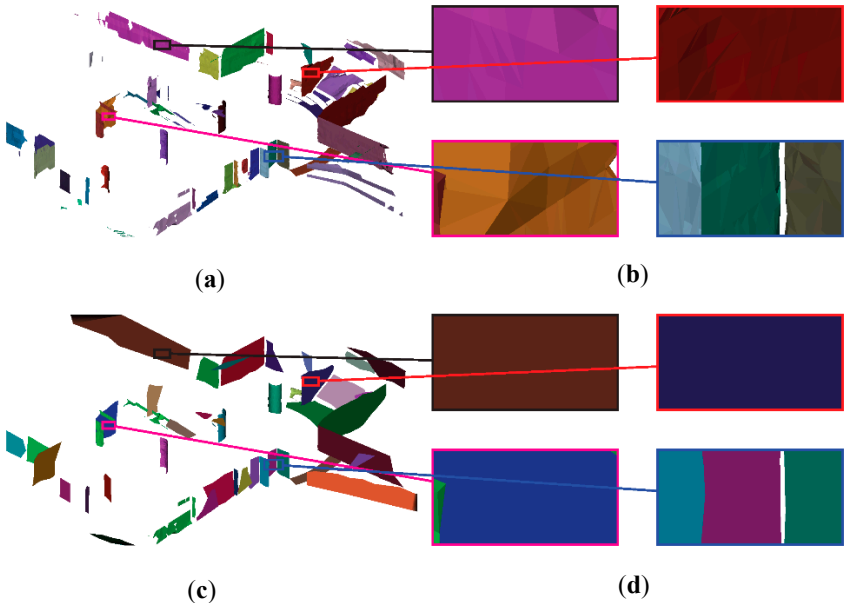


Figure 18. Surface reconstruction of a single frame point cloud without regularization. (**a**,**b**) show surface reconstruction based on alpha shape without plane regularization for comparison. (**c**,**d**) show the result after plane regularization. We can see that creases are flatter.

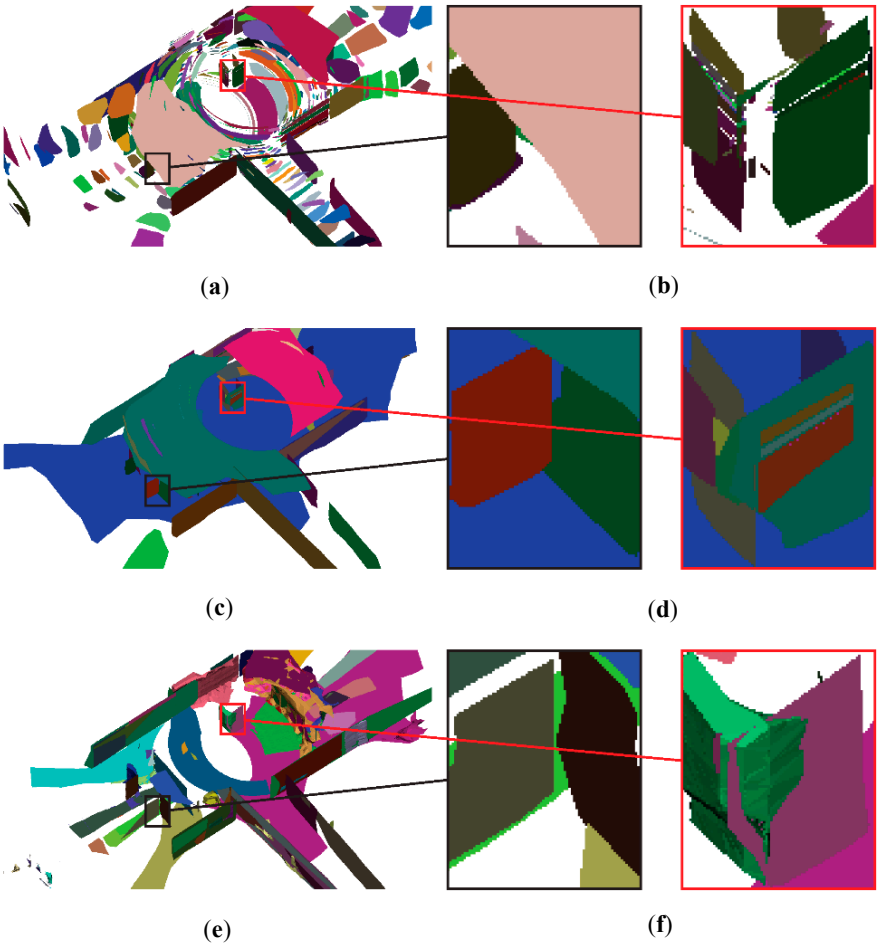


Figure 19. Surface reconstruction results for a multiple point cloud of LiDAR. Different colors of the plane indicate over-segmentation. (**a**,**b**) Region Growing; (**c**,**d**) IRIS Region Growing; (**e**,**f**) IRIS SLCC.

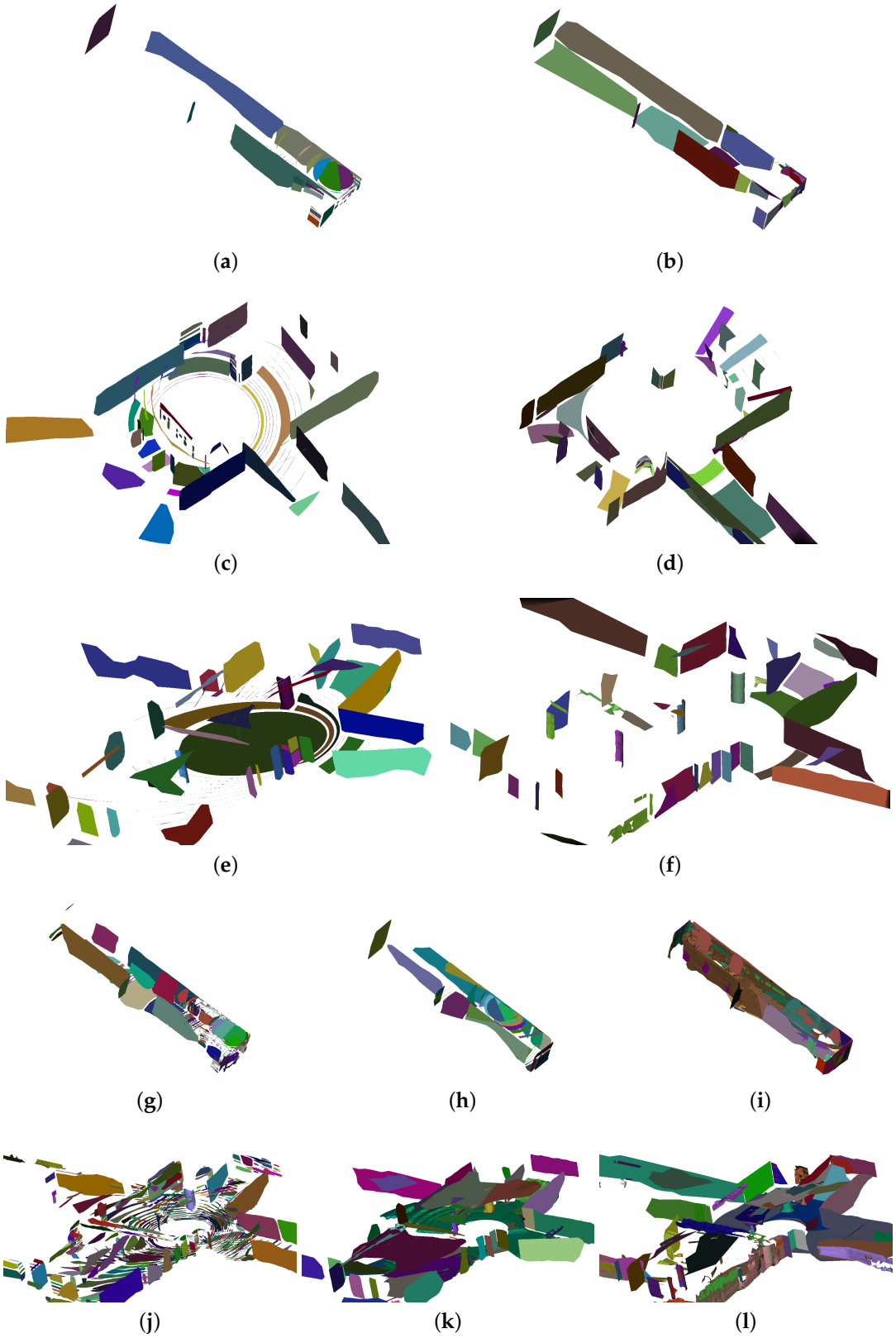


Figure 20. (**a**–**l**) More surface reconstruction results on three datasets. (**a**,**c**,**e**) Results using Region Growing for one frame on three datasets; (**b**,**d**,**f**) results using SLCC for one frame on three datasets; (**g**–**l**) results using Region Growing, IRIS Region Growing and IRIS SLCC for the Corridor and Entrance datasets.

7.5. Limitations

Scanlines from two different objects that are very close may not be appropriately separated with the proposed SLCC method. For instance, the scanline of the left foot is agglomerated into the segment of the ground as shown in Figure 16a and the wheels of the bicycle are also agglomerated into the segment of the ground as shown in Figure 16e,f. On the other hand, the object with a wide range is prone to being over-segmented. For example, the ground is over-segmented into several segments as shown in Figures 15c and 17e. Both of these two situations mostly occur on the segmentation related to the ground. A preprocessing for detecting and removing of the ground plane with methods such as RANSAC can be applied to improve the segmentation result.

8. Conclusions and Future Work

In this work, we present an optimized segmentation method, SLCC, customized for sparse and non-uniform mobile LiDAR data and an incremental scheme, IRIS, which can be adapted for any segmentation algorithm. SLCC segments the point cloud based on the continuity of consecutive points in terms of distance and direction. We utilize an adaptive threshold and take into account the incident angle to help overcome challenges related to the sparsity and non-uniformity of the point cloud. This minimized the possibility of miss-segmentation and over-segmentation. The incremental scheme IRIS can be adapted for any segmentation algorithm to combine the segments of different point clouds. It can also be used to segment a series of continuous frames of the point cloud recursively. IRIS utilizes the previous segmented results, thereby increasing the efficiency of segmentation.

We evaluated the effectiveness of IRIS and the segmentation performance of SLCC for a single frame and that of IRIS SLCC for multiple frames of point clouds on three datasets of different scales. In order to obtain a comparative study, we also performed segmentation using Region Growing. In terms of the time consumption, our proposed IRIS scheme shows an increase in the efficiency of segmentation when the segmentation results of the previous point cloud are available. As for segmentation, Region Growing possesses very good generality for segmenting a variety of point clouds. However, Region Growing becomes prone to over- and under-segmentation for the point cloud of mobile LiDAR data due to the sparsity and non-uniformity of the data, while our proposed SLCC exhibits better segmentation results. Despite the increased density of the point cloud obtained by combining multiple frames, over-segmentation is observed using Region Growing. Interestingly, over-segmentation is greatly suppressed by implementing Region Growing with IRIS. As an application example, the automatically refined surface of each segment is reconstructed, which demonstrates the quality of the segmentation results. From the reconstructed surfaces of multiple frames of point clouds, over-segmentation of planar objects by all three segmentation methods can be seen. This is caused by the registration error and can be improved by making the drift of registration smaller.

LiDAR devices are often used together with RGB panoramic cameras. Our future research efforts will focus on two subjects. One is the derivation of the calibration matrix between LiDAR and the panoramic camera. The correspondence of segmentation in both a 3D point cloud and a 2D image can be used to find the solution. The other is automatic texture extraction from panoramic images and automatic mapping onto the reconstructed surfaces that can be generated by the method proposed in this work.

Author Contributions: Weimin Wang proposed and implemented the algorithm, collected the data and performed the experiment, and wrote the manuscript. Weimin Wang and Ken Sakurada contributed to the improvement of the algorithm and the discussion of the results. Ken Sakurada and Nobuo Kawaguchi supervised the entire process of the research and reviewed the manuscript.

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

References

1. Nakagawa, M.; Kataoka, K.; Yamamoto, T.; Shiozaki, M.; Ohhashi, T. Panoramic rendering-based polygon extraction from indoor mobile LiDAR data. *ISPRS Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2014**, *40*, 181–186.
2. Previtali, M.; Barazzetti, L.; Brumana, R.; Scaioni, M. Towards automatic indoor reconstruction of cluttered building rooms from point clouds. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2014**, *2* , 281–288.
3. Rabbani, T.; van den Heuvel, F.A.; Vosselman, G. Segmentation of point clouds using smoothness constraint. *ISPRS Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2006**, *36*, 248–253.
4. Fischler, M.A.; Bolles, R.C. Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **1981**, *24*, 381–395.
5. Rusu, R.B.; Cousins, S. 3D is here: Point Cloud Library (PCL). In Proceedings of the IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011.
6. Grant, W.S.; Voorhies, R.C.; Itti, L. Finding planes in LiDAR point clouds for real-time registration. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, 3–7 November 2013.
7. Schnabel, R.; Wahl, R.; Klein, R. Efficient RANSAC for point-cloud shape detection. *Comput. Graph. Forum* **2007**, *26*, 214–226.
8. Oehler, B.; Stueckler, J.; Welle, J.; Schulz, D.; Behnke, S. Efficient multi-resolution plane segmentation of 3D point clouds. In *Intelligent Robotics and Applications*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 145–156.
9. Wang, M.; Tseng, Y.H. Incremental segmentation of lidar point clouds with an octree-structured voxel space. *Photogramm. Rec.* **2011**, *26*, 32–57.
10. Papon, J.; Abramov, A.; Schoeler, M.; Worgotter, F. Voxel cloud connectivity segmentation—supervoxels for point clouds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013.
11. Moosmann, F.; Pink, O.; Stiller, C. Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion. In Proceedings of the IEEE Intelligent Vehicles Symposium, Xi'an, China, 3–5 June 2009.
12. Zhang, J.; Singh, S. LOAM: Lidar odometry and mapping in real-time. In Proceedings of the Robotics: Science and Systems X, Berkeley, CA, USA, 12–17 July 2014.
13. Douillard, B.; Underwood, J.; Kuntz, N.; Vlaskine, V.; Quadros, A.; Morton, P.; Frenkel, A. On the segmentation of 3D LIDAR point clouds. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 9–13 May 2011.
14. Spinello, L.; Arras, K.O.; Triebel, R.; Siegwart, R. A layered approach to people detection in 3D range data. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, Atlanta, GA, USA, 11–15 July 2010.
15. Armeni, I.; Sener, O.; Zamir, A.R.; Jiang, H.; Brilakis, I.; Fischer, M.; Savarese, S. 3D semantic parsing of large-scale indoor spaces. In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, Cancún, Mexico, 4–8 December 2016.
16. Bosse, M.; Zlot, R.; Flick, P. Zebedee: Design of a spring-mounted 3-D range sensor with application to mobile mapping. *IEEE Trans. Robot.* **2012**, *28*, 1104–1119.
17. Corso, N.; Zakhor, A. Indoor localization algorithms for an ambulatory human operated 3D mobile mapping system. *Remote Sens.* **2013**, *5*, 6611–6646.
18. Oesau, S.; Lafarge, F.; Alliez, P. Planar shape detection and regularization in Tandem. *Comput. Graph. Forum* **2015**, *35*, 203–215.
19. Monszpart, A.; Mellado, N.; Brostow, G.J.; Mitra, N.J. RAPter: Rebuilding Man-made Scenes with Regular Arrangements of Planes. *ACM Trans. Graph.* **2015**, *34*, 103:1–103:12.
20. Whelan, T.; Ma, L.; Bondarev, E.; de With, P.; McDonald, J. Incremental and batch planar simplification of dense point cloud maps. *Robot. Auton. Syst.* **2015**, *69*, 3–14.
21. Biswas, J.; Veloso, M. Planar polygon extraction and merging from depth images. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012.

22. Wang, J.; Shan, J. Segmentation of LiDAR point clouds for building extraction. In Proceedings of the American Society for Photogramm. Remote Sens. Annual Conference, Baltimore, MD, USA, 8–13 March 2009; pp. 9–13.
23. Gopi, M.; Krishnan, S. A fast and efficient projection-based approach for surface reconstruction. In Proceedings of the 15th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI), Fortaleza-CE, Brazil, 7–10 October 2002.
24. Lee, D.T.; Schachter, B.J. Two algorithms for constructing a Delaunay triangulation. *Int. J. Comput. Inf. Sci.* **1980**, *9*, 219–242.
25. Chew, L.P. Constrained delaunay triangulations. *Algorithmica* **1989**, *4*, 97–108.
26. Turner, E.; Zakhor, A. Watertight planar surface meshing of indoor point-clouds with voxel carving. In Proceedings of the International Conference on 3D Vision, Seattle, WA, USA, 29 June–1 July 2013.
27. Ma, L.; Favier, R.; Do, L.; Bondarev, E.; de With, P.H.N. Plane segmentation and decimation of point clouds for 3D environment reconstruction. In Proceedings of the IEEE 10th Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, 11–14 January 2013.
28. Mura, C.; Mattausch, O.; Villanueva, A.J.; Gobbetti, E.; Pajarola, R. Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. *Comput. Graph.* **2014**, *44*, 20–32.
29. Zhang, Z. Iterative point matching for registration of free-form curves and surfaces. *Int. J. Comput. Vis.* **1994**, *13*, 119–152.
30. Segal, A.; Haehnel, D.; Thrun, S. Generalized-ICP. In Proceedings of the Robotics: Science and Systems V, Seattle, WA, USA, 28 June–1 July 2009.
31. Yu, F.; Xiao, J.; Funkhouser, T. Semantic alignment of LiDAR data at city scale. In Poceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
32. 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.
33. Edelsbrunner, H.; Kirkpatrick, D.; Seidel, R. On the shape of a set of points in the plane. *IEEE Trans. Inf. Theory* **1983**, *29*, 551–559.
34. Edelsbrunner, H.; Mücke, E.P. Three-dimensional alpha shapes. In Proceedings of the 1992 Workshop on Volume Visualization (VVS), Boston, MA, USA, 19–20 October 1992; Association for Computing Machinery (ACM): New York, NY, USA, 1992.
35. The Visualization ToolKit (VTK). Available online: http://www.vtk.org (accessed on 16 November 2016).
36. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The KITTI dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237.