# Out-of-core Outlier Removal for Large-scale Indoor Point Clouds

Linlin Ge
State Key Lab of CAD&CG, Zhejiang University
Hangzhou, China
linlinge@zju.edu.cn

Jieqing Feng
State Key Lab of CAD&CG, Zhejiang University
Hangzhou, China
jqfeng@cad.zju.edu.cn

## Abstract

**An out-of-core outlier removal method for large-scale indoor point clouds is proposed, which takes into account point density distribution, geometric shape information, and appropriate thresholds in outlier removals. First, a low-resolution point cloud (LPC) is obtained using random downsampling. It has the same density distribution as the raw point clouds (RPC), which is important information for outlier removal. The correspondences from the LPC to the RPC are also recorded. The outliers in the LPC are removed via a global threshold. The outliers in the RPC are roughly removed guided by the cleaned LPC. Then, the cleaned LPC is segmented into planar parts and non-planar parts; and the LPC segmentation is transferred to the RPC. Finally, the outliers in each RPC segment are removed elaborately via a local threshold by exploring the shape information. The experiments show that the proposed method can improve the quality of outlier removal results.**

## 1. Introduction

Raw point clouds (RPCs) from a laser scanner or 3D reconstruction algorithm will inevitably contain numerous outliers owing to light conditions, measurement errors, and so on. Such outliers can significantly impact subsequent processing operations, such as denoising, point cloud clustering, segmentation, and 3D surface reconstruction. Furthermore, in indoor scene applications, outliers can affect the calculation accuracy of geometric information of buildings. Indoor point cloud outlier processing is challenging owing to the large quantity and space size involved. Thus, efficient and intelligent outlier removal has become an essential preprocessing step in indoor point cloud processing.

As mentioned above, the primary difficulty in processing an indoor point cloud is its large scale. Raw data may contain points up to a few hundred millions or more; thus, loading them into the main memory is difficult. Out-of-core technology can solve this problem. Stream processing [34] and hierarchy processing [37, 40, 12] are two types of typical approaches. Unfortunately, few out-of-core approaches are utilized for outlier removal. Outlier processing has the following characteristics: point density distribution and geometric shapes are essential information that must be preserved in the out-of-core processing procedure.

Compared to a scanned hand-sized object, the size of an indoor point cloud is much larger. Finding a suitable threshold for the entire point cloud outlier removal is difficult. Researchers proposed different strategies for processing large point clouds. For example, to obtain reliable thresholds, Sotoodeh [42] proposed a hierarchical outlier removal method, in which a Euclidean minimum spanning tree and Gabriel graph at the top and bottom layers are established, respectively. Points within a small cluster size and with large spacing from other clusters are removed. However, the method's memory consumption in processing a massive point cloud is exceedingly high. In recent years, with the development of deep learning, several learning-based methods [36, 43] were proposed to learn models and automatically determine optimal thresholds from training data. However, such methods require vast amounts of training data to achieve satisfactory results, and preparing training data is also a challenge.

To address the above issues, an out-of-core outlier removal method is proposed in this paper, which is capable of utilizing point density distribution and geometric shapes and determining suitable thresholds for large-scale data. In the proposed method, the random sampling (RS) strategy capable of maintaining the point density distribution is utilized to extract the low-resolution point cloud (LPC) from the RPC. Simultaneously, the point correspondences between the LPC and the RPC are also recorded. The LPC,

instead of the RPC, is loaded into the memory; thus, memory consumption is reduced significantly. The operations on the LPC can be transferred to the RPC according to the recorded correspondences. To process the indoor point cloud elaborately, the outlier removal procedure is divided into two phases. In the first phase, the outliers in the LPC are removed with the proposed outlier removal I (OR-I) method. Next, the OR-I result for the LPC is transferred to the RPC with the help of the recorded correspondences and a distance constraint. In the second phase, a divide-and-conquer strategy is utilized to further remove the outliers in RPC. Firstly, the cleaned LPC is segmented into two types of segments: planar segment and non-planar segment. Then, the corresponding segments in RPC are also obtained via the recorded correspondences. According to the characteristics of the two types of segments, the plane structure information in the planar segments and the point density information in the non-planar segments are adopted to remove the outliers, respectively. Since the segments of the RPC are processed sequentially, an appropriate threshold can be determined for each RPC segment, meanwhile memory consumption is reduced. Benefiting from the divide-and-conquer strategy, the multi-threshold mechanism for outlier removal in the RPC can lead to accurate results. The experiments on synthetic and real-world data show that the proposed out-of-core outlier removal method can achieve more elaborate outlier removal results, with low memory consumption, compared with previous methods.

The contributions of this study are summarized as follows:

- An out-of-core method maintaining the point density distribution is explored. Point density distribution is one of the essential information in outlier removal. To maintain the point density distribution, random sampling is employed to extract the LPC from the RPC in the out-of-core procedure. As the LPC is lightweight, loading it into the main memory is feasible, and analyzing the outliers of the RPC through the LPC is also possible.

- A hierarchical structure, containing the LPC and RPC layers, is proposed to utilize geometric shapes sufficiently for outlier removal. First, the LPC is segmented into two types of segments: planar segment and non-planar segment. Then, the segmentation of the LPC is transferred to the RPC, and the related geometric shape information in the LPC is also transferred to the RPC for the subsequent outlier removal procedures.

- A multi-threshold mechanism is proposed to improve outlier removal accuracy. First, a method called OR-I is applied to the LPC using a global threshold. Then, the cleaned LPC via OR-I is used as a guide model to roughly remove the outliers in the RPC by setting a distance constraint from the cleaned LPC to the RPC. Second, each RPC segment is processed using its threshold for improving the outlier processing accuracy.

The rest of this paper is organized as follows. The related works are introduced in Section 2, and the proposed out-of-core processing method is presented in Section 3.1. The global outlier removal method and model-based refinement outlier removal method are described in Section 3.2 and Section 3.3, respectively. Experimental results for synthetic and real-world data and discussion are given in Section 4, and finally, the conclusions are drawn and future work is indicated in Section 5.

## 2. Related Works

This section introduces several works related to our study, specifically, on out-of-core technologies and outlier removal methods.

### 2.1. Out-of-core Point Cloud Processing

The term "out of core" typically refers to processed data that are too large to be loaded into the main memory. Out-of-core technology tries to obtain the one-to-one mapping between the data in the main memory and the data in the external memory. According to their work mechanisms, out-of-core technologies for massive point cloud processing can be classified into stream processing and hierarchical processing.

**Stream Processing.** This method was first proposed by Renato Pajarola [34] and consists of two steps, that is, the preprocessing step and the stream-processing step. In the preprocessing step, the RPC is sorted into an orderly sequential structure in a Euclidean space using an external sorting technique [46]. The steam-processing step tries to process this extensive data in a specified sweeping direction. Although this method works well on local differential operators such as normal estimation, curvature estimation, smoothing, and so on, Renato Pajarola [34] pointed out that its memory consumption grows unproportionally when faced with small clustered outliers. In other words, this method will occupy a large main memory when faced with point clouds containing large amounts of small clustered outliers. Moreover, the batch processing flow makes the stream processing approach rarely utilize global geometric information, such as geometric shapes, for outlier removal. These two factors limit stream processing in outlier removal applications.

**Hierarchical Processing.** QSplat [37] is the first method that uses a hierarchical structure for the point-based rendering of large-scale point clouds. Numerous octree-based methods emerged thereafter, such as Potree [40], Entwine

[1], octree-based external memory mesh (OEMM) [12], and so on. Typically, such methods first establish an octree, in which each leaf node contains a subset of the raw data, stored continuously in the external memory. The data in each leaf node are essentially raw data with no duplicates. For a specific operation, only the required nodes are loaded into the main memory through the octree. Although these hierarchical methods achieve satisfactory effects in many applications, such as point-based rendering [11, 27, 16, 6, 41], simplification [26, 12, 11, 27], shape editing [34, 38], surface reconstruction [7], iso-surface extraction [28], and so on, few studies discussed its application to outlier removal. Previous hierarchical works partitioned RPC in a Euclidean space, which will destroy the geometric shape information of the RPC. In this study, a hierarchical approach for maintaining the geometric shapes in each hierarchy is first explored for outlier removal.

### 2.2. Point Cloud Outlier Removal Processing

Outliers can be classified into sparse isolated outliers (SIOs), dense isolated outliers (DIOs), and non-isolated outliers (NIOs) [42, 47]. An illustration of each type of outlier is presented in Figure 1. The traditional method generally uses a single criterion to remove outliers from a 3D point cloud. Recently, combined methods using multiple criteria and learning-based techniques became the mainstream approaches.

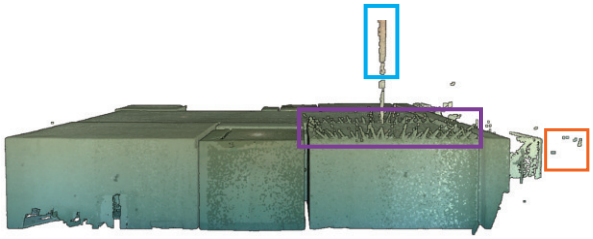


Figure 1. Illustration of SIOs (orange rectangle), DIOs (blue rectangle), and NIOs (purple rectangle)

**Single criterion methods.** Weyrich et al. [48] proposed a nearest-neighbor reciprocity criterion that believes that the valid point may be in the outlier's neighborhood. The outlier is highly unlikely to be in the valid point's neighborhood. The local outlier factor (LOF) [9], local correlation integral (LOCI) [35], and local outlier probability (LoOP) [23] score an outlier based on each point's local density, and the outliers are the points with high outlier scores. The cluster-based local outlier factor (CBLOF) [19] regards small-clustered points far from the other points as outliers. Campos et al. [10] used a quadratic function to fit the local neighboring points. Then a RANSAC-based procedure is performed to detect the outliers which are far away from the best fitting model. Single criterion methods are well-suited for one type of outliers but perform poorly for other types of outliers.

**Combined methods.** To overcome the shortcomings of single criterion methods, numerous works classified outliers into different types and removed them using different criteria. Ning et al. [33] proposed two geometric criteria for removing SIOs and NIOs. However, the authors indicated the method's failure in removing DIOs. To remove DIOs efficiently, S. Sotoodeh [42] proposed a hierarchical outlier removal method that constructs two graphs to remove isolated outliers and NIOs. Wang et al. [47] proposed an outlier removal method that first converts NIOs into isolated outliers then removes the isolated outliers in the second stage. Meanwhile, Ge et al. [15] developed a type-based outlier removal framework. By considering the characteristics of a point cloud, the framework classified outliers elaborately, making the processing algorithm highly targeted. Although such combined methods achieve promising results for small point clouds, most of them can not directly solve the outlier removal problem for large-scale indoor point clouds in an out-of-core way owing to their high memory costs.

**Learning-based methods.** Rakotosaona et al. [36] proposed a supervised network called PointCleanNet, and Stucker et al. [43] developed a method based on a random forest classifier that incorporates predefined semantic classes into the learning process to improve precision. Unfortunately, both methods require the loading of the entire point cloud into the main memory, making them unsuitable for processing point clouds with hundreds of millions of points.

## 3. Out-of-core Outlier Removal for Large-scale Indoor Point Clouds

As the size of the scanned or reconstructed indoor point cloud exceeds the main memory's acceptable range, our objective is to remove the outliers with low memory consumption. Figure 2 illustrates the proposed out-of-core outlier removal method pipeline, which contains three phases

The out-of-core structure construction phase aims to establish a mapping from the external data to the in-core data. For this purpose, one LPC (see Figure 2[b]) capable of being loaded into the main memory is first obtained using random sampling (RS). Next, the cleaned LPC (see Figure 2[c]) is obtained using the proposed OR-I method. Then, the entire cleaned LPC is loaded into the main memory, and the KD-tree of the cleaned LPC is constructed. The KD-tree will enable each point in the RPC to search its nearest neighbor in the cleaned LPC. The correspondences between the RPC and cleaned LPC are encoded as an external-memory-stored hash table. A detailed description of the out-of-core structure construction phase is presented in Section 3.1.

The global outlier removal phase tries to roughly remove the outliers in the RPC according to the stored hash table and cleaned LPC. Some outliers in the RPC (see Figure 2[e]) are removed through a distance threshold. A detailed

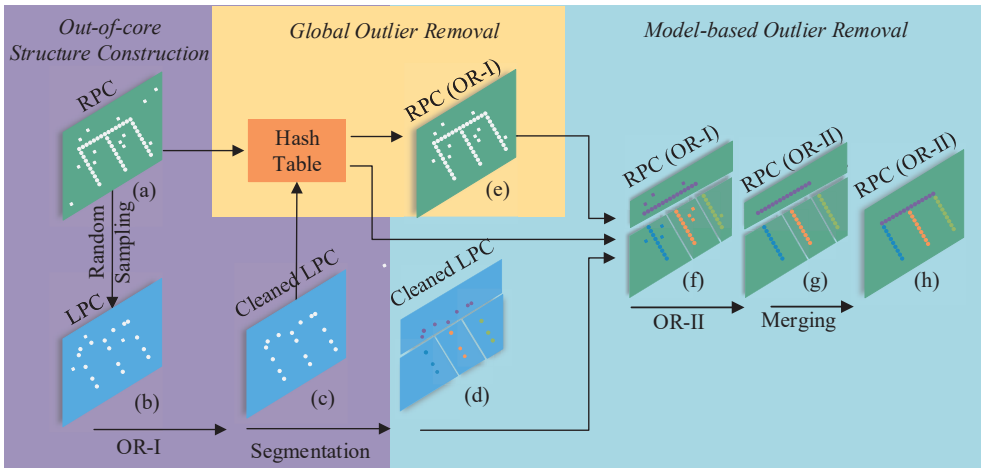



Figure 2. Pipeline of out-of-core outlier removal method for large indoor point cloud

description of the global outlier removal phase is presented in Section 3.2.

To further refine the global outlier removal results, the divide-and-conquer strategy is employed. The cleaned LPC is first segmented into serval geometric segments (see Figure 2[d]) by considering the geometric features, such as the position and normal, of each point. As the correspondences between the cleaned LPC and RPC are recorded in the hash table, segmenting the globally cleaned RPC (see Figure 2[e]) into several segments according to the cleaned LPC segmentation results is straightforward (see Figure 2[d]). The cleaned RPC segments are classified into two types, that is, planar segments and non-planar segments. Both segments are processed differently. The detailed description of the model-based outlier removal phase using the divide-and-conquer strategy is given in Section 3.3.

### 3.1. Out-of-core Structure Construction

#### 3.1.1 Downsampling

The downsampling phase generates an LPC from the RPC. The selected downsampling method should satisfy the following properties. As outlier removal methods are sensitive to the point density distribution, the selected downsampling methods should not change the point cloud density distribution. Meanwhile, the number of LPCs can be specified by the user, and it should be as fast as possible.

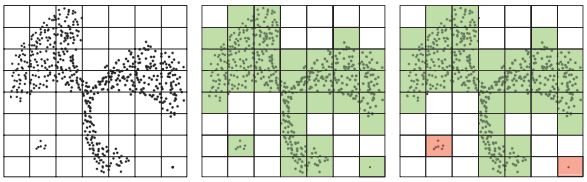
Hu et al. [21] verified the effectiveness and efficiency of random sampling (RS) compared to other downsampling approaches, namely, farthest point sampling (FPS), inverse density importance sampling (IDIS), generator-based sampling (GS), continuous relaxation sampling (CRS), and policy gradient-based sampling (PGS). Voxel sampling (VS) is also a commonly used downsampling approach. However, VS is slower than RS and cannot specify the LPC quantity. In summary, RS is more effective and efficient than FPS, IDIS, GS, CRS, PGS, and VS. Thus, RS is adopted to obtain the LPC from the RPC.

The RS implementation procedure is summarized into two steps. First, the subset indices of the RPC are generated. Let the number of points in the RPC and LPC be $n$ and $m$, respectively. Then, a random generator is utilized to generate an integer index set $S = \{s_1, \cdots, s_m\} \in [1, n]$ . According to Acharya et al. [4], this random generator keeps the final output point density distribution unchanged. Second, the points in the RPC with indices within the integer set $S$ are extracted to compose the LPC.

#### 3.1.2 Outlier Removal I (OR-I)

The outlier removal I phase aims to remove the outliers in the previously obtained LPC with minimal time consumption. As the LPC quantity is reasonable, most existing outlier removal methods can be adopted. Inspired by S. Sotoodeh [42] and Wang et al. [47], the NIOs, SIOs, and DIOs in the LPC are removed sequentially.

For NIOs, the OR-I method separates NIOs into isolated outliers. To this end, local outlier probability (LoOP) [23] is an ideal choice, as it can separate points with different local densities and generate usable results without frequently adjusting the parameters.



(a) Divide to cells (b) Get new cloud (c) Clustering

Figure 3. Schematic diagram for dealing with isolated outliers (red cell)

For SIOs and DIOs, cluster-based methods can achieve satisfactory performance. However, the time complexity of

most clustering algorithms is linearly proportional to the input data quantity; thus, such algorithms become extremely slow when handling millions of points. Inspired by the statistical information grid-based clustering method (STING) [14], a fast cell-based outlier removal method is described subsequently. Figure 3 illustrates the idea of the method. First, the bounding box of the input point cloud is divided into cuboid cells (see Figure 3[a]). Second, a set of centroids of points in each cell can form a point cloud (see Figure 3[b]). Finally, the cluster with a quantity prominently lower than that of the other clusters and that is far from the others is classified as an outlier cluster, in which all the points are outliers (see Figure 3[c]). In this way, the SIOs and DIOs in the LPC can be removed effectively.

#### 3.1.3 Indexing

The OR-I-cleaned LPC obtained in the previous step can offer the point density distribution and geometric information for us in eliminating the outliers in the RPC. However, the correspondences between the OR-I-cleaned LPC and RPC are missing. Let $\boldsymbol{I}_R$ and $\boldsymbol{I}_L$ be the point index sets of the RPC and OR-I-cleaned LPC, respectively. An example of the hash table storing the mapping from $\boldsymbol{I}_L$ to $\boldsymbol{I}_R$ is shown in Figure 4. The hash table is established through two phases. In the first phase, the KD-tree of the OR-I-cleaned LPC is constructed. In the second phase, for each point $\boldsymbol{p}_i$ in the RPC, its nearest neighbor $\boldsymbol{q}_j$ in the OR-I-cleaned LPC is found using the KD-tree.

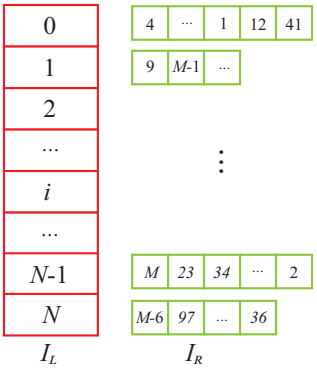



Figure 4. Hash table storing the index mapping between $\boldsymbol{I}_L$ of OR-I-cleaned LPC and $\boldsymbol{I}_R$ of RPC.

As a result, maintaining the hash table in the current main memory is feasible. If the point cloud quantity is one billon, then the hash table only requires approximately 700 MB main memory.

### 3.2. Global Outlier Removal

As the hash table established the correspondences between the cleaned LPC and RPC, some outliers in the RPC can be removed using a distance threshold. Let $\boldsymbol{p}_i$ be a point in the cleaned LPC. According to the hash table, its corresponding point set in the RPC are $\{\boldsymbol{q}_i^j | j = 1, \cdots, h\}$. The distance of point $\boldsymbol{p}_i$ and $\boldsymbol{q}_i^j$ is $D(\boldsymbol{p}_i, \boldsymbol{q}_i^j)$ . If the distance $D(\boldsymbol{p}_i, \boldsymbol{q}_i^j)$ is larger than a specified threshold $\tau_m$, then point $\boldsymbol{q}_i^j$ will be regarded as an outlier. The pseudocode of the global outlier removal method is presented as follows:

**Algorithm 1** Global Outlier Removal

**Input:** Cleaned LPC, Hash table
**Output:** Cleaned RPC

```
for each point p_i in cleaned LPC do
    Querying p_i's corresponding point set {q_i^j | j = 1, ..., h} in hash table
    Calculating the distance D(p_i, q_i^j)
    if D(p_i, q_i^j) > τ_m then
        removing point q_i^j
    end if
end for
```

### 3.3. Model-based Outlier Removal

#### 3.3.1 Geometric Segmentation

In this step, the cleaned LPC is segmented into simple geometric parts. The segmentation on the cleaned LPC will be transferred to the RPC. The outliers in each RPC segment are removed further using its geometric characteristics. The point cloud segmentation for the cleaned LPC is first formulated as a graph partition problem, as follows:

Given an undirected weighted graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{F}, \boldsymbol{E}, \boldsymbol{\Omega})$, where $\boldsymbol{V}$ is the set of nodes, for each node $\boldsymbol{v}_i \in \mathbb{R}^3$ in $\boldsymbol{V}$ is associated with a feature $\boldsymbol{f}_i \in \mathbb{R}^d$. $\boldsymbol{F}$ is the feature set, and $\boldsymbol{E}$ is the set of edges. Each edge in $(\boldsymbol{v}_i, \boldsymbol{v}_j) \in \boldsymbol{E}$ has a weight $\omega_{ij} \in \boldsymbol{\Omega}$. In this study, a k-nearest neighbor graph[25] is used to construct the graph for the cleaned LPC. The weight of each edge is set as follows:

$$w_{ij} = \frac{1}{0.5 + d_{i,j}/\bar{d}} \tag{1}$$

where $d_{i,j}$ is the distance of edge $(\boldsymbol{v}_i, \boldsymbol{v}_j)$, and $\bar{d}$ is the mean distance of all the edges. The goal is to embed the feature space into a label space $\boldsymbol{L} = \{\boldsymbol{l}_1, \cdots, \boldsymbol{l}_i, \cdots\}$ by minimizing the following energy function.

$$\arg\min_{\boldsymbol{L}} \{D_{data} + \mu D_{smooth}\} \tag{2}$$

where $\mu$ is the regularization strength that determines the coarseness of the resulting segments. It should be positive and set 0.1 in this study. $D_{data}$ and $D_{smooth}$ are the data term item and smooth item, respectively, which are defined as follows:

$$\begin{cases} D_{data} = \sum_{i=1}^{|\boldsymbol{V}|} \|\boldsymbol{l}_i - \boldsymbol{f}_i\|^2 \\ D_{smooth} = \sum_{(\boldsymbol{v}_i, \boldsymbol{v}_j) \in \boldsymbol{E}} \omega_{ij} [\boldsymbol{l}_i - \boldsymbol{l}_j \neq 0] \end{cases} \tag{3}$$

where $[\cdot]$ is the Iverson bracket. Although Equation (3) is a nonconvex optimization, the $l_0$-cut pursuit algorithm [24] can quickly find an approximate solution. Consequently, points with similar labels are considered to be in the same cluster.

In the energy function (2), the normal of the points is adopted as feature. Therefore, it will be estimated first. Two crucial issues should be taken into account in the normal estimation, that is, sharp feature preservation and normal consistency. An indoor scene contains many sharp features, such as wall corners, edges, and so on. If such sharp features are not considered, then the segmentation results will be inaccurate. In recent years, numerous interesting works [29, 30] achieved satisfactory results regarding this issue. Boulch et al. [8] proposed a normal estimation method based on Hough transform to deal with sharp features. To demonstrate its advantages, this study compares the segmentation results of a point cloud in which normal is estimated through principal component analysis (PCA) and Hough transform (Figure 5). Figure 5(a) shows that the segmentation via PCA tends to be smooth at the corner and edge, which is an over-segmentation result. However, the normal estimated via Hough transform can lead to a satisfactory segmentation, as shown Figure 5(b).

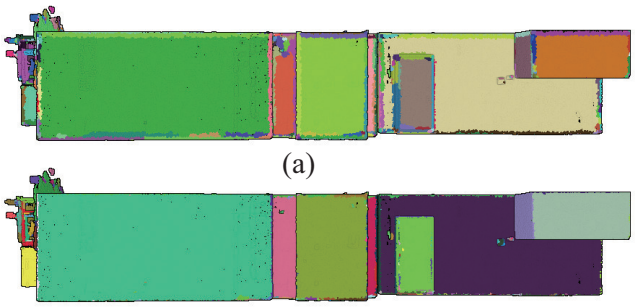


Figure 5. Segmentation results using different normal estimation methods; (a) normal estimated via PCA; (b) normal estimated via Hough transform

Although the normal estimation method based on the Hough transform works well on sharp features, it cannot guarantee the consistency of the normal results. This inconsistency will also lead to over-segmentation. For example, in Figure 6(a), the normal orientations are not consistent in the red rectangle area, leading to a false segmentation result. To make the estimated normal consistent, the minimum spanning tree (MST) approach [20] is adopted. The approach establishes a minimum spanning tree based on a neighbor graph and starts from one or more vertices with known normal orientations then flips the normal by traversing the spanning tree, if necessary. Figure 6(b) shows an improved segmentation result via the MST approach.

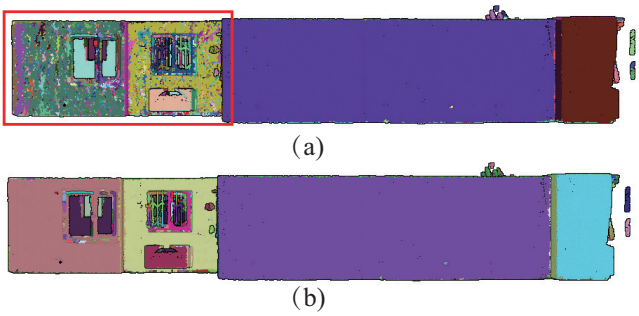


Figure 6. Segmentation results; (a) input point cloud without normal consistency operation; (b) input point cloud with normal consistency operation

### 3.3.2 Outlier Removal II (OR-II)

In the OR-I phase, the cleaned LPC is obtained. Moreover, a distance threshold from the LPC to the RPC is introduced to roughly remove the outliers in the global outlier removal phase (Section 3.2).

The OR-II method aims to further refine the rough global outlier removal result by considering the planar geometry in the RPC. The OR-II method takes the segments obtained in the previous step as the input and outputs cleaned segments. To fully utilize the prior planar information of the indoor scene, all the planar segments must be identified. Suppose $l_1$, $l_2$, and $l_3$ are the lengths of the axis-aligned bounding box of a segment. As the far outliers in the OR-I-cleaned RPC are removed in the global outlier removal phase, then the planar segments can be determined by the following conditions:

$$\begin{cases} l_1 \leq l_2 \leq l_3 \\ l_1 < \tau_1 \\ l_2 > \tau_2 \\ l_3 > \tau_3 \end{cases} \tag{4}$$

$\tau_1$, $\tau_2$, and the $\tau_3$ are thresholds for $l_1$, $l_2$, and $l_3$, respectively. Through Equation (4), the segments can be classified into two types, namely, planar and non-planar segments. The OR-II method will process the RPC segments sequentially. Then all the cleaned segments will be loaded sequentially and merged into one single file.

**Planar Segment Outlier Removal**

For each point $\boldsymbol{p}_i$ in a planar segment, its k-nearest neighbors are denoted as $\left\{\boldsymbol{p}_i^j | j = 1, \cdots, k\right\}$. Note that $\boldsymbol{p}_i^1 = \boldsymbol{p}_i$, because $\boldsymbol{p}_i$ is the nearest point to itself. Let $\bar{\boldsymbol{p}}_i$

be the centroid of the neighborhood. The covariance matrix $\boldsymbol{C} \in \mathbb{R}^{3\times3}$ of its $k$-nearest neighborhood is defined as

$$\boldsymbol{C} = \frac{1}{k}\begin{bmatrix} \boldsymbol{p}_i^1 - \bar{\boldsymbol{p}}_i \\ \cdots \\ \boldsymbol{p}_i^k - \bar{\boldsymbol{p}}_i \end{bmatrix}^T \begin{bmatrix} \boldsymbol{p}_i^1 - \bar{\boldsymbol{p}}_i \\ \cdots \\ \boldsymbol{p}_i^k - \bar{\boldsymbol{p}}_i \end{bmatrix} \tag{5}$$

Suppose that the eigenvalues of the matrix $\boldsymbol{C}$ are $\lambda_1, \lambda_2, \lambda_3$, and they all satisfy the condition $\lambda_1 \leq \lambda_2 \leq \lambda_3$. Then, the surface variation $\sigma_k(\boldsymbol{p}_i)$ is defined as follows:

$$\sigma(\boldsymbol{p}_i) = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3} \tag{6}$$

According to our perception, the surface variation should approximate zero in a planar segment. In other words, the region with a high surface variation is likely to be an outlier. Thus, the surface variation is adapted to detect the outliers in the planar segments. With Equation (6), the surface variations of each point can be obtained. The surface variations at each point in each planar segment are denoted as $\boldsymbol{\sigma} = \{\sigma(\boldsymbol{p}_i)|i = 1, \cdots, n\}$. Then, the outlier detection rule Tukey's fences [45] is utilized to detect the anomalous surface variations. Let $Q_1$ and $Q_3$ be the lower and upper quartiles of $\boldsymbol{\sigma}$, respectively, and point $\boldsymbol{p}_i$ is regarded as an outlier with the following condition.

$$\sigma(\boldsymbol{p}_i) \geq Q_3 + \alpha(Q_3 - Q_1) \tag{7}$$

where $\alpha$ is a threshold, and the larger the threshold, the more the points believed to be outliers. The reasonable range of $\alpha$ is within $(0, 50]$, which is set to 5 in this study.

**non-planar Segment Outlier Removal**

Compared with the planar-like segments, the non-planar segments have more complicated geometric shapes, and their geometric prior information is harder to use. Among the outlier removal methods, density-based methods [9, 23, 35] are widely adopted when faced with arbitrarily distributed point clouds, because they have no specific point cloud distribution requirements. LoOP [23] has two advantages among the density-based methods: (i) it is robust when dealing with the nonuniform point cloud, and (ii) it works well without needing to make complex parameter adjustments. Therefore, LoOP is adopted to remove the outliers in the non-planar segments. LoOP takes non-planar segments as the input and outputs the outlier probability score for each point in the non-planar segment. In this study, a point with an outlier probability score larger than 0.8 is regarded as an outlier.

## 4. Experiments

All the experiments were performed on a 3.7 GHz Intel Core™ i9-10900X desktop with a 128-GB memory. To comprehensively evaluate the proposed method, the experiment consists of three parts. First, the rationality of the chosen downsampling strategy is verified in Section 4.1. Second, the memory usage and runtime of the proposed out-of-core method are compared with those of other approaches in Section 4.2. Third, the comparison results of the different outlier removal methods dealing with massive indoor point clouds are presented in Section 4.3.

In this study, three large-scale point cloud datasets were utilized to evaluate the proposed method. Semantic3D [18] is a dataset of natural scenes with over 4 billion points in total. Owing to the large quantity of each point cloud in Semantic3D, it was utilized to test the validation of the proposed out-of-core method. The Stanford large-scale indoor spaces (S3DIS) [5] is a clean scanned indoor point cloud dataset. As it is clean, this dataset was adopted to generate the synthetic data to evaluate the proposed outlier removal method. Three real-world models (i.e., Houses A, B, and C) were scanned with a structure light scanner and adopted to further verify the effect of the proposed outlier removal method on real applications.

### 4.1. Validation of Downsampling Strategy

#### 4.1.1 Runtime Comparison

Hu et al. [21] compared RS with FPS [32], (IDIS) [17], GS [13], (CRS) [3], and PGS [49] and pointed out that RS works best when dealing with large-scale point clouds. However, the authors did not consider VS. In this section, RS and VS are compared. To make the comparison reasonable, the experiments were conducted on different data scales containing $10^7$, $10^8$, and $10^9$ points. As mentioned above, VS cannot specify the amount of the generated LPC. To compare the runtimes, the cell size of VS was specified, and the quantity of its corresponding LPC was determined accordingly. In this way, RS was performed under the same LPC quantity specification as VS. Table 1 shows the cell size of the VS approach for each raw data.

Table 1. Cell size of VS for each raw data.

| Quantity of RPC | Cell Size (m) | Quantity of LPC |
|---|---|---|
| $10^7$ | 0.01003660 | 5,001,666 |
| $10^8$ | 0.01300075 | 5,346,474 |
| $10^9$ | 0.01200075 | 6,982,502 |

Most downsampling algorithms perform two steps. In the first step, the algorithms generate the indices of the LPC. In the second step, based on the indices, the algorithms will extract the LPC from the raw data. To avoid ambiguity, both steps were included in the runtime. The runtime and memory consumption results of VS and RS are presented in Table 2. The table shows that RS used less memory and

runtime than VS when performing on hundreds of millions of point clouds or more.

Table 2. Runtime (sec) and memory consumption (GB) comparison between VS and RS.

| Time / Memory | VS (sec / GB) | RS (sec / GB) |
|---|---|---|
| $10^7$ | **21** / 0.90 | 26 / **0.70** |
| $10^8$ | 54 / 6.04 | **25** / **0.73** |
| $10^9$ | 574 / 59.64 | **339** / **0.93** |

#### 4.1.2 Density Distribution Preservation

Point density distribution is an important factor for outlier removal. Therefore, the point density distribution of the LPC should be approximate to that of the raw data. Before the point density distribution is discussed, the point density must first be defined. For the convenience of the statistical point density, the point density of a point is defined as the number of neighbor points within a specified radius. By counting the number of each point density, the point density distribution can be obtained.

To verify the effectiveness of RS in maintaining the point density distribution, the point density distributions of the RPC, the LPC (RS result), and the VS result are compared. A comparison is shown in Figure 7. The point cloud generated via RS (see Figure 7[b]) has a point density distribution similar to that of the raw data (see Figure 7[a]), whereas the point density distribution of the point cloud generated via VS (see Figure 7[c]) deviates from that of the raw data.

### 4.2. Evaluation of Data Structure in Out-of-core Method

As mentioned above, Semantic3D is adopted to evaluate the data structure in the proposed out-of-core method. Potree [40] is one of the most common octree-based out-of-core methods. Hence, Potree and the in-core KD-tree method are implemented and compared with the proposed out-of-core method.

Table 3. Runtime and memory consumption in constructing the out-of-core data structure

| Scenes | Numbers | Time (s) / Memory (GB) | | |
|---|---|---|---|---|
| | | In-core | Potree | Ours |
| sg27_1 | 322M | 100/19.28 | 650/4.11 | **462/3.09** |
| sg27_2 | 496M | 149/31.53 | 959/**3.73** | **685**/4.17 |
| sg27_3 | 422M | 129/26.83 | 869/4.52 | **587/3.69** |
| sg27_4 | 280M | 91/17.87 | 470/**1.91** | **404**/2.79 |
| sg27_5 | 218M | 73/13.90 | 364/**1.99** | **323**/2.45 |
| House A | 116M | 57/9.33 | 172/1.79 | **149/1.74** |
| House B | 553M | 277/49.68 | 818/**2.12** | **599**/4.05 |
| House C | 827M | 534/75.22 | 1285/**2.12** | **877**/5.53 |

The implementation of the out-of-core technology consisted of two steps, that is, the data structure construction step and the mapping step. For a fair comparison, both steps are considered in the experiments. The statistics of runtime and memory consumption are presented in Table 3.

Compared with the in-core results, it is observed that the Potree and the proposed method have significantly decreased memory consumption but take a considerable amount of time to construct the data structure, as shown in Table 3. Compared with the Potree, the proposed method reduces runtime by 23.39% but increases memory consumption by 35.86%. Therefore, compared with Potree, the proposed method trades space for time. In addition, the proposed method has the advantage of keeping the point density distribution of the LPC consistent with the RPC.

### 4.3. Evaluation of Outlier Removal

To evaluate the proposed outlier removal method, some classical methods, including Reciprocity [48], LoOP, and SOR, are implemented for comparison. SOR and Reciprocity are implemented via a point data abstract library (PDAL) [2], and LoOP is provided by MeshLab. For the quantitative evaluation, the $F$-score [22] is used to evaluate the accuracy and completeness of the outlier removal results.

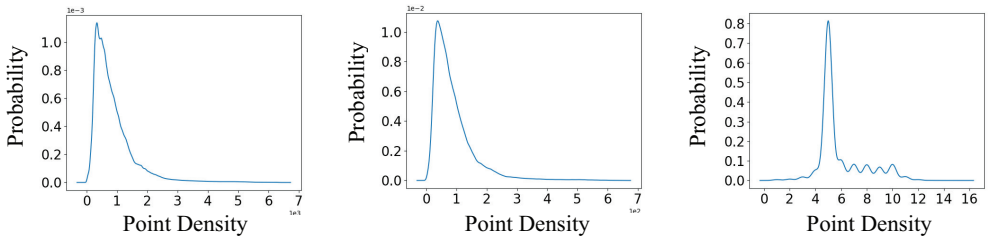
#### 4.3.1 Evaluation on S3DIS

As mentioned previously, S3DIS [5] is a clean scanned indoor point cloud dataset. S3DIS contains six large-scale indoor scenes. The point clouds are clean and regarded as the ground truth. Inspired by Rakotosaona et al. [36], Gaussian noise was added to the S3DIS data to generate the RPCs containing outliers. Next, different outlier removal methods were applied to the RPCs. The $F$-score [22] is a commonly used indicator for measuring the accuracy and completeness of the results comprehensively. Therefore, the F-score is adopted to provide quantitative comparisons among the outlier removal results. The larger the $F$-score, the better the performance of the method. As shown in Table 4, the proposed method achieved the best F-score on the synthetic S3DIS dataset.

Table 4. Quantitative results ($F$-score values) of different outlier removal methods (Num: Number of point in RPC, Rec: Reciprocity, Raw: the RPC with Gaussian Noise)

| Scenes | Num | Raw | Rec | LoOP | SOR | Ours |
|---|---|---|---|---|---|---|
| Area_1 | 48M | 97.08 | 99.18 | 98.92 | 98.73 | **99.73** |
| Area_2 | 104M | 97.08 | 98.27 | 97.08 | 98.86 | **99.24** |
| Area_3 | 41M | 97.05 | 98.27 | 98.80 | 98.83 | **99.15** |
| Area_4 | 96M | 97.10 | 98.25 | 97.10 | 98.88 | **99.81** |
| Area_5 | 173M | 97.07 | 92.15 | 97.07 | 98.73 | **99.81** |
| Area_6 | 44M | 97.09 | 99.16 | 98.88 | 98.73 | **99.70** |

Figure 8 shows a visual comparison among the different outlier removal methods performed on the scenes Area 1 -




(a) Raw (b) Random Sampling (c) Voxel Sampling

Figure 7. Comparison of point density distribution between RS and VS downsampling approaches.

3. Two perspective views from the outside and inside viewpoints are presented for the scenes Area 1 - 3. It can be seen that SOR and the proposed method worked better than Reciprocity and LoOP. Furthermore, the proposed method had a better effect on the NIO removal than the SOR method.

#### 4.3.2 Evaluation on Real-world Scenes

In real applications, outliers cannot be avoided despite careful scanning operations. In addition, unlike scanning single objects, scanning indoor point clouds would be uneven owing to their extensive scanning range. Here, three real-world models scanned by an industrial scanner are adopted to verify the effect of the proposed outlier removal method on real-world applications.

In Figure 9, the first row and the second row are the outside and inner perspective views for House A, respectively. Reciprocity, LoOP, and SOR fail to remove the DIOs (red circles). In addition, some SIOs (green circles) remain in the Reciprocity and LoOP results. The point density in the yellow circle is smaller than that in the other areas, leading to SOR removing regular points by mistake. Among these methods for House A, the proposed method performs best.

House B is another scene in Figure 9. The point cloud in the red circle and rectangle contained some DIOs. Except for the proposed method, the other methods are unable to deal with the DIOs effectively. The inner perspective view of House B showed that Reciprocity and LoOP failed to remove some SIOs in the green circle. SOR removed some regular points (yellow circle) by mistake. Similar to House A, the proposed method also achieves the best performance.

Similar to Houses A and B, the proposed method performs best again when processing the DIOs (red circles and rectangles) and SIOs (green circles) in House C. Moreover, LoOP achieved the worst results, as it removed some regular points (yellow circle) by mistake.

In addition, Table 5 provides the runtime and memory consumption of the different methods when dealing with real-world scenes. It can be seen that LoOP and SOR have a shorter runtime than Reciprocity and the proposed methods. However, when processing House C, memory consumptions of LoOP and SOR are greater than 90 GB which limits their applications. Compared with Reciprocity, the proposed method used less time and memory. Overall, the proposed method is more suitable for processing large-scale point clouds.

Table 5. Runtime and memory consumption of different methods for real-world scenes (Num: Number of point in RPC, Rec: Reciprocity)

| Scenes | Num | Time (h) / Memory (GB) | | | |
|---|---|---|---|---|---|
| | | Rec | LoOP | SOR | Ours |
| A | 116M | 1.85/9.4 | **0.02**/13.7 | 0.04/11.8 | 0.22/**0.7** |
| B | 553M | 3.24/9.2 | 0.08/22.2 | **0.04**/19.6 | 0.47/**1.0** |
| C | 827M | 11.28/62 | **0.14**/95.9 | 0.28/93.3 | 1.80/**3.4** |

## 5. Conclusion and Future Work

An out-of-core outlier removal method is proposed for processing large-scale indoor data, which is capable of preserving the point density distribution, utilizing geometric shapes, and determining suitable thresholds simultaneously. To preserve the point density distribution, RS is introduced into the out-of-core procedure to extract the LPC from the RPC. As the LPC is lightweight, loading it into the main memory is feasible, and analyzing the outliers of the RPC through the LPC is also possible. The geometric shapes are important information for outlier removal. To take full advantage of this information, the LPC is segmented into two types of segments, namely, planar and non-planar segments. Then, the segmentation of the LPC is transferred to the RPC, and the related geometric shape information in the LPC is also transferred to the RPC for the subsequent outlier removal procedures. To determine suitable thresholds, a multi-threshold mechanism is proposed in this paper. First, the method called OR-I is applied to the LPC using the global threshold. Then, the cleaned LPC via the global threshold in the OR-I method is used as a guide model to roughly remove the outliers in the RPC by setting a distance constraint from the cleaned LPC to the RPC. Second,

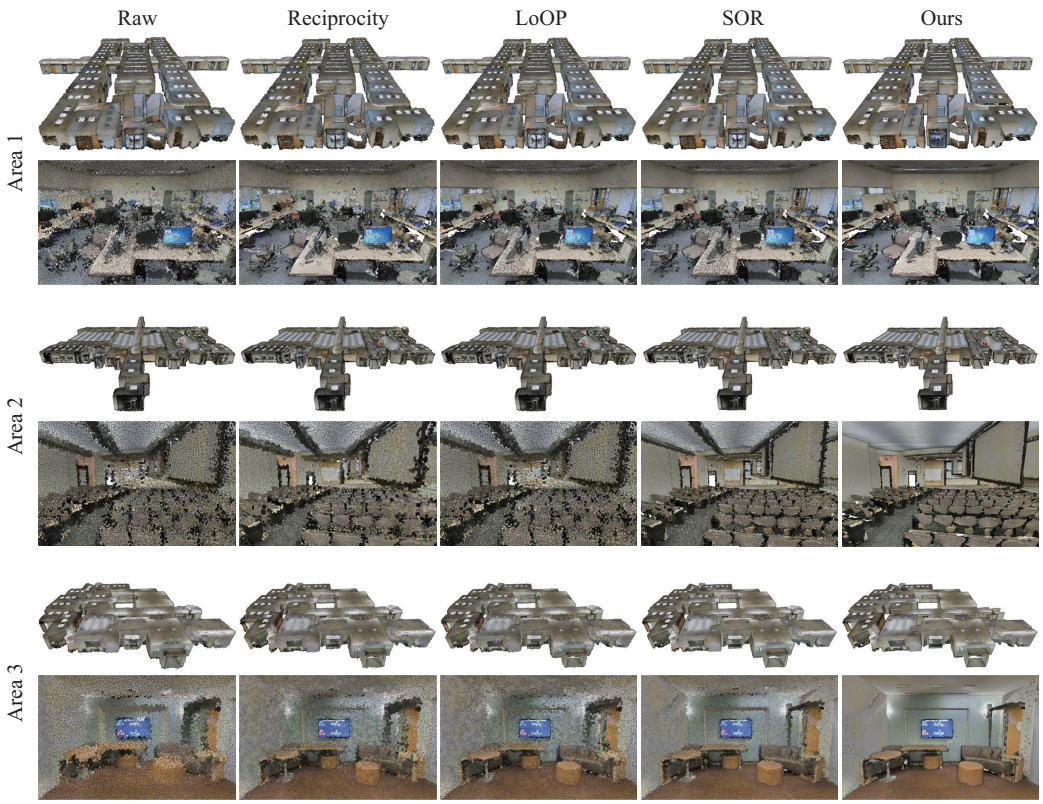



Figure 8. Visual comparison among outlier removal methods for the scenes Area 1 - 3; the first row and second row of each Area present the outside and inner perspectives of the results

each segment in RPC is processed individually by using its threshold for improving the outlier processing accuracy. According to the final experiments, the proposed out-of-core method does not require an external sorting procedure, thereby making it faster than its competitors. Moreover, the outlier removal test results showed that the proposed outlier removal method obtained highly accurate results.

In the OR-II phase, the plane prior in the planar segment is the decisive factor for the outliers. However, for non-planar segment, the geometric prior of the underlying surface is not considered. Therefore, the proposed method is suitable for scenes with a large number of planar structures. Hence, a future research direction is to consider other regular primitives [39] in removing outliers for complex geometric structures.

Moreover, the proposed method still contains several complex parameters, which can cause two problems. (i) The outlier removal method is sensitive to parameter adjustments. Hence, different point clouds depend on different parameters. (ii) Some of the parameters are not adequately intuitive. Inspired by several papers [44, 50, 31], a self-determined parameter algorithm will be appealing in future research.

## 6. Acknowledgement

The authors wish to acknowledge Yuliang Sun, Peng Yao, Shan Luo, Lei Wang, and Qitong Zhang. The discussions with them inspired us immensely. We also sincerely acknowledge the authors and institutions who provided the testing datasets, including the S3DIS and Semantic3D.

## References

[1] Entwine. https://github.com/connormanning/entwine, 2010. 3

[2] Pdal point data abstraction library. https://doi.org/10.5281/zenodo.2556738, 2018. 8

[3] A. Abid, M. F. Balin, and J. Y. Zou. Concrete autoencoders for differentiable feature selection and reconstruction. arXiv:1901.09346, 2019. 7

[4] A. S. Acharya, A. Prakash, P. Saxena, and A. Nigam. Sampling: Why and how of it. *Indian Journal of Medical Specialties*, 4(2):330–333, 2013. 4

[5] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese. Joint 2d-3d-semantic data for indoor scene understanding. arXiv:1702.01105, 2017. 7, 8

[6] J. Baert, A. Lagae, and P. Dutré. Out-of-core construction of sparse voxel octrees. In *Proceedings of the High-Performance Graphics Conference*, pages 27–32, 2013. 3

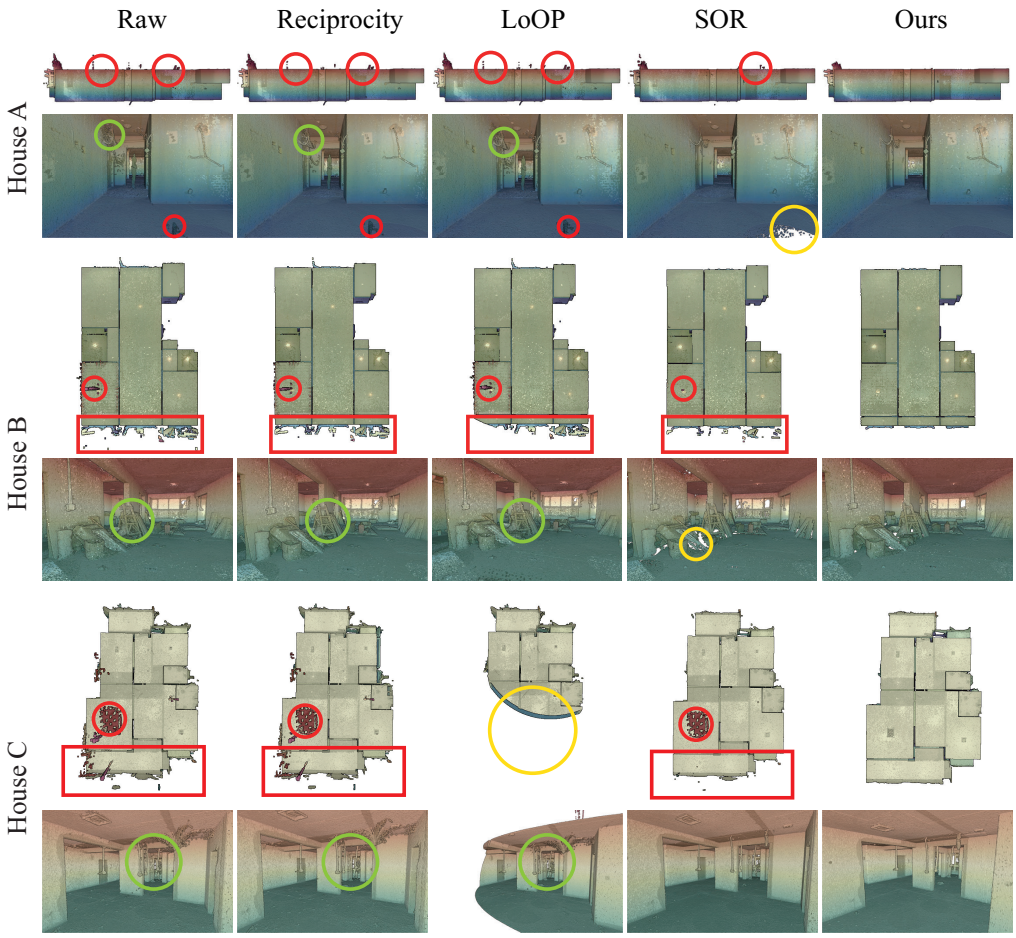



Figure 9. Qualitative comparison on outlier removal methods for Houses A, B, and C; the first row and second row present the outside and inner perspectives of the results, respectively


[7] M. Bolitho, M. Kazhdan, R. Burns, and H. Hoppe. Multilevel streaming for out-of-core surface reconstruction. In *Proceeding of the Symposium on Geometry Processing*, pages 69–78, 2007. 3

[8] A. Boulch and R. Marlet. Fast and robust normal estimation for point clouds with sharp features. *Computer Graphics Forum*, 31(5):1765–1774, 2012. 6

[9] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *Proceedings of The ACM SIGMOD International Conference on Management of Data*, pages 93–104, 2000. 3, 7

[10] R. Campos, R. García, P. Alliez, and M. Yvinec. Splat-based surface reconstruction from defect-laden point sets. *Graphical Models*, 75(6):346–361, 2013. 3

[11] Y.-J. Chiang, J. El-Sana, P. Lindstrom, R. Pajarola, and C. T. Silva. Out-of-core algorithms for scientific visualization and computer graphics. In *IEEE Visualization*, pages 35–48, 2003. 3

[12] P. Cignoni, C. Montani, C. Rocchini, and R. Scopigno. External memory management and simplification of huge meshes. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):525–537, 2003. 1, 3

[13] O. Dovrat, I. Lang, and S. Avidan. Learning to sample. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2760–2769, 2019. 7

[14] G. Gan, C. Ma, and J. Wu. *Data clustering: theory, algorithms, and applications*. Society for Industrial and Applied Mathematics, 2020. 5

[15] L. Ge and J. Feng. Type-based outlier removal framework for point clouds. *Information Sciences*, 580:436–459, 2021. 3

[16] E. Gobbetti, F. Marton, and J. A. I. Guitián. A single-pass gpu ray casting framework for interactive out-of-core rendering of massive volumetric datasets. *The Visual Computer*, 24(7):797–806, 2008. 3

[17] F. Groh, P. Wieschollek, and H. P. Lensch. Flex-convolution. In *Proceeding of the Springer Asian Conference on Computer Vision*, pages 105–122, 2018. 7

[18] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys. Semantic3d.net: A new large-scale point cloud classification benchmark. In *ISPRS annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 91–98, 2017. 7

[19] Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003. 3

[20] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, pages 71–78, 1992. 6

[21] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham. Randla-net: efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020. 4, 7

[22] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4):1–13, 2017. 8

[23] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Loop: local outlier probabilities. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 1649–1652, 2009. 3, 4, 7

[24] L. Landrieu and G. Obozinski. Cut pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs. *SIAM Journal on Imaging Sciences*, 10(4):1724–1766, 2017. 6

[25] L. Landrieu and M. Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018. 5

[26] P. Lindstrom. Out-of-core simplification of large polygonal models. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, pages 259–262, 2000. 3

[27] P. Lindstrom. Out-of-core construction and visualization of multiresolution surfaces. In *Proceedings of the Symposium on Interactive 3D Graphics*, pages 93–102, 2003. 3

[28] R. U. Lobello, F. Dupont, and F. Denis. Out-of-core adaptive iso-surface extraction from binary volume data. *Graphical Models*, 76(6):593–608, 2014. 3

[29] D. Lu, X. Lu, Y. Sun, and J. Wang. Deep feature-preserving normal estimation for point cloud filtering. arXiv:2004.1156, 2020. 6

[30] X. Lu, S. Schaefer, J. Luo, L. Ma, and Y. He. Low rank matrix approximation for 3D geometry filtering. arXiv:1803.06783, 2018. 6

[31] X. Lu, S. Wu, H. Chen, S.-K. Yeung, W. Chen, and M. Zwicker. GPF: GMM-inspired feature-preserving point set filtering. *IEEE Transactions on Visualization and Computer Graphics*, 24(8):2315–2326, 2018. 10

[32] C. Moenning and N. Dodgson. Fast marching farthest point sampling. Technical report, University of Cambridge, 2003. 7

[33] X. Ning, F. Li, G. Tian, and Y. Wang. An efficient outlier removal method for scattered point cloud data. *PloS One*, 13(8):e0201280, 2018. 3

[34] R. Pajarola. Stream-processing points. In *IEEE Visualization Conference*, pages 239–246, 2005. 1, 2, 3

[35] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. Loci: fast outlier detection using the local correlation integral. In *Proceedings of the IEEE International Conference on Data Engineering*, pages 315–326, 2003. 3, 7

[36] M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N. J. Mitra, and M. Ovsjanikov. Pointcleannet: Learning to denoise and remove outliers from dense point clouds. *Computer Graphics Forum*, 39(1):185–203, 2020. 1, 3, 8

[37] S. Rusinkiewicz and M. Levoy. Qsplat: a multiresolution point rendering system for large meshes. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, pages 343–352, 2000. 1, 2

[38] C. Scheiblauer and M. Wimmer. Out-of-core selection and editing of huge point clouds. *Computers & Graphics*, 35(2):342–351, 2011. 3

[39] R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, 2007. 10

[40] M. Schütz. *Potree: Rendering large point clouds in web browsers*. PhD thesis, Vienna University of Technology, 2015. 1, 2, 8

[41] M. Schütz, S. Ohrhallinger, and M. Wimmer. Fast out-of-core octree generation for massive point clouds. *Computer Graphics Forum*, 39(7):155–167, 2020. 3

[42] S. Sotoodeh. Hierarchical clustered outlier detection in laser scanner point clouds. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(3/W52):383–388, 2007. 1, 3, 4

[43] C. Stucker, A. Richard, J. D. Wegner, and K. Schindler. Supervised outlier detection in large-scale mvs point clouds for 3d city modeling applications. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4:263–270, 2018. 1, 3

[44] L. Trujillo, U. Lopez, and P. Legrand. Soap: Semantic outliers automatic preprocessing. *Information Sciences*, 526:86–101, 2020. 10

[45] J. W. Tukey et al. *Exploratory data analysis*, volume 2. Addison Wesley Publishing Company, 1977. 7

[46] J. S. Vitter. External memory algorithms and data structures: Dealing with massive data. *ACM Computing Surveys*, 33(2):209–271, 2001. 2

[47] Y. Wang and H.-Y. Feng. Outlier detection for scanned point clouds using majority voting. *Computer-Aided Design*, 62:31–43, 2015. 3, 4

[48] T. Weyrich, M. Pauly, R. Keiser, S. Heinzle, S. Scandella, and M. H. Gross. Post-processing of scanned 3d surface data. In *Proceedings of the Symposium on Point Based Graphics*, pages 85–94, 2004. 3, 8

[49] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning*, pages 2048–2057, 2015. 7

[50] D. Zhang, X. Lu, H. Qin, and Y. He. Pointfilter: Point cloud filtering via encoder-decoder modeling. *IEEE Transactions on Visualization and Computer Graphics*, 27(3):2015–2027, 2020. 10