applied sciences

MDPI

Article

Dynamic Downsampling Algorithm for 3D Point Cloud Map Based on Voxel Filtering

Wenqi Lyu [1], Wei Ke [1,*], Hao Sheng [2], Xiao Ma [1] and Huayun Zhang [3]

[1] Faculty of Applied Sciences, Macao Polytechnic University, Macao 999078, China; wenqi.lyu@mpu.edu.mo (W.L.); xiao.ma@mpu.edu.mo (X.M.)
[2] State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China; shenghao@buaa.edu.cn
[3] Faculty of Intelligent Technology and Engineering, Chongqing University of Science and Technology, Chongqing 401331, China
* Correspondence: wke@mpu.edu.mo

Abstract: In response to the challenge of handling large-scale 3D point cloud data, downsampling is a common approach, yet it often leads to the problem of feature loss. We present a dynamic downsampling algorithm for 3D point cloud maps based on an improved voxel filtering approach. The algorithm consists of two modules, namely, dynamic downsampling and point cloud edge extraction. The former adapts voxel downsampling according to the features of the point cloud, while the latter preserves edge information within the 3D point cloud map. Comparative experiments with voxel downsampling, grid downsampling, clustering-based downsampling, random downsampling, uniform downsampling, and farthest-point downsampling were conducted. The proposed algorithm exhibited favorable downsampling simplification results, with a processing time of 0.01289 s and a simplification rate of 91.89%. Additionally, it demonstrated faster downsampling speed and showcased improved overall performance. This enhancement not only benefits productivity but also highlights the system's efficiency and effectiveness.

Keywords: voxel filtering; 3D point cloud downsampling; dynamic downsampling; improved voxel filtering

Citation: Lyu, W.; Ke, W.; Sheng, H.; Ma, X.; Zhang, H. Dynamic Downsampling Algorithm for 3D Point Cloud Map Based on Voxel Filtering. *Appl. Sci.* **2024**, *14*, 3160. https://doi.org/10.3390/app14083160

Academic Editor: Antonio Fernández-Caballero

Received: 27 February 2024
Revised: 10 March 2024
Accepted: 12 March 2024
Published: 9 April 2024

1. Introduction

With the rapid development of hardware devices for acquiring 3D point cloud data, such as lidar [1] and depth cameras [2], the number of 3D point cloud data that can be acquired is getting larger and larger. Three-dimensional point cloud data are widely used in autonomous driving [3–5], robots [6], augmented virtual reality [7], and smart cities [8,9]. More and more attention is being paid to research on 3D point cloud technology. In the field of smart home, equipment such as sweeping robots [10] and robot housekeepers [11] need to construct 3D point cloud maps of indoor scenes. However, a 3D point cloud map in an indoor environment alone has hundreds of millions of points. Processing such a huge point cloud map, therefore, requires a lot of computing resources. The downsampling of 3D point cloud data plays a key role in subsequent operations, such as segmentation [12], classification [13], and target recognition of 3D point cloud maps [14–17].

Point cloud downsampling methods can be categorized into deep learning-based [18–20], grid-based [21,22], clustering-based [23], and voxel-based downsampling methods [24]. Although, in recent years, researchers have made significant progress in point cloud downsampling, the challenges of detail loss and parameter tuning remain unresolved.

To address these issues in 3D point cloud downsampling, we investigate the downsampling algorithms for 3D point clouds and propose a dynamic downsampling algorithm for 3D point cloud maps based on voxel filtering. This algorithm aims to retain the edge information of point clouds. We introduce two modules: the dynamic downsampling

module and the point cloud edge extraction module. The former dynamically segments 3D point cloud maps to perform adaptive voxel downsampling, while the latter preserves the edge information in 3D point cloud maps. We conduct comparative experiments with other downsampling algorithms, demonstrating the superior simplification effect of our proposed downsampling method.

The structure of the paper is organized as follows: In Section 2, we provide an overview of the background and significance of downsampling 3D point clouds. Section 3 introduces the principle of voxel downsampling and highlights the challenges associated with voxel downsampling. Section 4 presents the dynamic downsampling algorithm for 3D point cloud maps based on voxel filtering. In Section 5, we conduct comparative experiments between our downsampling algorithm and other existing methods, demonstrating the superior simplification effect of our approach. Finally, Section 7 summarizes the innovative aspects and contribution of this research paper and identifies its limitation.

2. Related Work

Point cloud downsampling is a crucial step in point cloud processing, as it effectively reduces the size of point cloud data, decreases computational load, and accelerates subsequent operations. Point cloud downsampling methods can be categorized into deep learning-based, grid-based, clustering-based, and voxel-based downsampling.

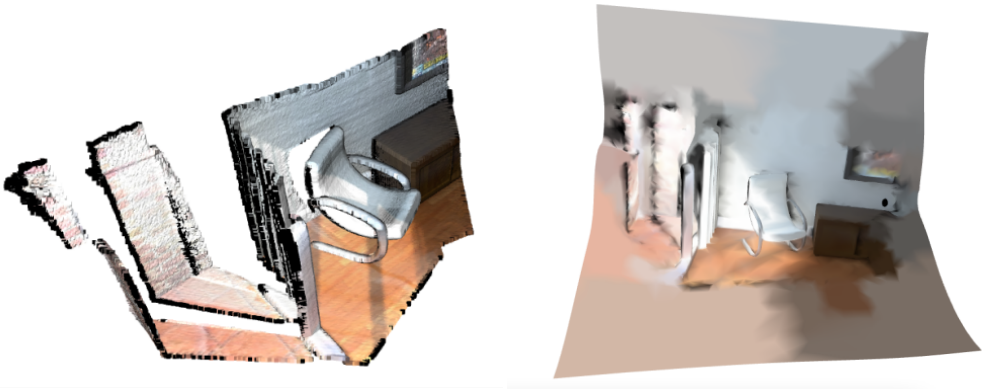
2.1. Deep Learning-Based Downsampling

The application of downsampling methods based on deep learning in point cloud processing is relatively scarce. Many combine deep learning technology with traditional machine learning methods. Yu [18] proposed SIEV-Net, using a height information supplementary module to minimize the height information loss during the aggregation of point features in the voxel network. He [19] proposed a sparse voxel map attention network, SVGA-Net, using a voxel map module and a sparse-to-density regression module to achieve comparable 3D detection tasks from original lidar data with a good simplification rate. Que [20] proposed VoxelContext-Net for static and dynamic point cloud compression. Nguyen [25] proposed a learning-based static point cloud geometric downsampling method that exploited a deep convolutional neural network with a mask to learn the probability distribution of voxels. Qin [26] applied deep learning methods to point cloud downsampling and proposed a Gaussian model voxel network—GVnet. It introduces a lightweight convolutional neural network to learn point cloud representation and semantic information while using part of the point cloud information to improve the efficiency and accuracy of point cloud sampling. FoldingNet, proposed by Yangyan Li [27], is a point cloud downsampling method based on autoencoders. Gezawa [28] used a laser point cloud downsampling method based on deep convolutional autoencoders to build a sampling module based on a combined hybrid model. Point cloud and voxel data are used to determine the relationship between each pair of point voxels. In this model, each voxel is embedded using the magnitude view, which is the Euclidean distance between the view and the center of the object, as well as the angle between each pair of views.

2.2. Grid-Based Downsampling

Grid-based downsampling methods [21,22] are another class of commonly used downsampling methods, first proposed by Garland and Heckbert [29]. The principle is to grid the point cloud space and then use the average or weight of the points in each grid to replace all the points in the original grid, thus reducing the number of points. Yuan [30] proposed a point cloud simplification algorithm based on voxelized grid downsampling. This method divides the point cloud space into different subspaces, then samples in each subspace, and finally combines them into a complete sampled point cloud. This method is efficient and scalable, with the capability to cope with large-scale point cloud sampling problems. In experiments, this method showed better sampling effects and faster calculation speed than other point cloud sampling methods. Zhang [31] proposed an adaptive triangular

mesh model redistribution algorithm that could preserve the details and overall shape of the point cloud at the same time. Grid subsampling fits the point cloud plane, but it is easy to blur and stretch the shape of the original 3D point cloud image. The comparison before and after grid subsampling is shown in Figure 1.

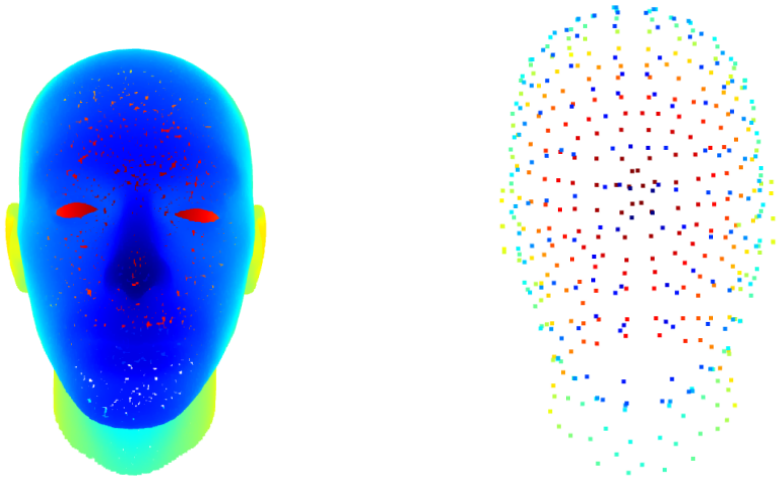


(**a**) (**b**)

Figure 1. Comparison of 3D point clouds before and after grid downsampling. (**a**) Original point cloud image. (**b**) Grid downsampling.

2.3. Clustering-Based Downsampling

Clustering downsampling was not a main research direction in the early days but a technique applied to specific problems. With the popularization and application of point cloud data, clustering downsampling has become one of the common technologies in the field of point cloud processing and has also been more researched and discussed. The principle is to divide the point cloud into several clusters with a clustering algorithm and then sample those clusters. Chao [23] proposed a clustering method based on K-means. This method clusters the original majority class samples into the same number of clusters as the minority class samples through K-means clustering, then finds the sample center for each cluster, and uses the sample center as the new majority class sample. The algorithm can handle irregular sampling points and is robust against changes in sampling point density. Clustering downsampling has a poor sampling effect on high-density point clouds. It is easy to lose the feature points of the point cloud image and blur the edge information. A comparison of clouds before and after clustering downsampling is shown in Figure 2.

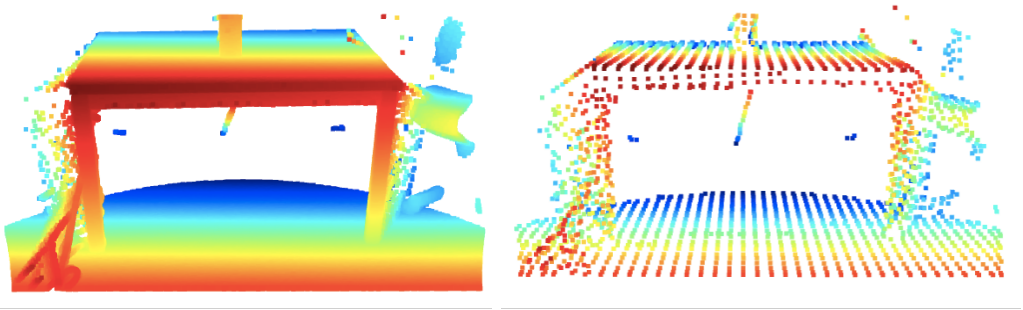


(**a**) (**b**)

Figure 2. Comparison of 3D point clouds before and after cluster downsampling. (**a**) Original point cloud image. (**b**) Point cloud image after cluster downsampling.

2.4. Voxel-Based Downsampling

Voxel-based downsampling is a commonly used downsampling method first proposed by Rusinkiewicz and Levoy [32] in 2001. The principle of voxel downsampling is to downsample the original point cloud by putting the 3D point cloud data into a 3D voxel grid, calculating the center of gravity within each voxel, and then using the center of gravity as the new sampling point. The advantage of voxel downsampling is the ability to convert original point cloud data into regular grid data, so that point cloud data can be processed and analyzed more conveniently [33]. At the same time, voxel downsampling can also adjust the accuracy and density of downsampling by changing parameters such as the voxel size and step size. This method is able to handle point cloud data sizes larger by orders of magnitude. Domestic and foreign researchers have also proposed many research methods based on voxel downsampling. Xiao [24] proposed a point cloud downsampling method based on hierarchical voxel segmentation, which balances point cloud downsampling by increasing the number of segmentation layers. A comparison of 3D point clouds before and after voxel downsampling is shown in Figure 3.



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

Figure 3. Comparison of 3D point clouds before and after voxel downsampling. (**a**) Original point cloud image. (**b**) Voxel downsampling.

In summary, while each of these downsampling methods has unique advantages, as shown in Table 1, they also present significant limitations, especially when dealing with the complexities of 3D point cloud data. Traditional methods like grid and cluster downsampling excel in noise reduction and feature preservation but often struggle with dense or irregularly distributed datasets, leading to potential loss of critical details. On the other hand, voxel-based downsampling provides a straightforward approach to regulate sampling density and maintain the overall shape and structure of point clouds. However, it tends to overlook fine local details and may introduce noise, underscoring the necessity for a more dynamic and adaptable solution.

Table 1. Advantages and disadvantages of various downsampling methods.

Methods	Advantages	Disadvantages
Deep learning-based downsampling	It can handle large-scale point cloud data efficiently and has fast processing speed.	It requires a significant number of training data and abundant computational resources. In cases of imbalanced training samples, it may lead to overfitting.
Grid downsampling	(1) It can effectively remove noise and redundant information, yielding consistent sampling results. (2) Compared with voxel downsampling, it is better at preserving the local features of point clouds.	It is not effective for dense and unevenly distributed point clouds.

Table 1. *Cont.*

Methods	Advantages	Disadvantages
Cluster downsampling	(1) It can effectively eliminate noise and redundant information while preserving local details. (2) It performs better on point clouds with non-uniform distributions.	It performs poorly in processing high-density point clouds and is prone to creating voids.
Voxel downsampling	(1) Simple and easy to implement, its sampling density can be controlled by adjusting the voxel size. (2) It can preserve the overall shape and structural characteristics of the original point cloud.	It cannot handle local detail information and may introduce noisy points.

Given these considerations, our proposed dynamic downsampling algorithm based on voxel filtering emerges as a powerful alternative. It is specifically designed to address the limitations of existing methods by efficiently processing high-density, large indoor scenes and preserving both the edge features and local detail information of 3D point cloud maps. This approach not only enhances the downsampling performance but also significantly reduces the need for manual parameter tuning, setting a new benchmark for the field.

3. Preliminaries

Voxel downsampling is a method for reducing the density of point clouds while preserving their structural information. In voxel downsampling, p_i denotes the i^{th} point in a point cloud of n points, with coordinates (x_i, y_i, z_i). The process divides the cloud into cubic voxels of edge length l, assigning each point to a voxel with center coordinates (x_c, y_c, z_c). The floor function, $\lfloor \cdot \rfloor$, ensures that points are accurately grouped by rounding down their coordinates to the nearest voxel center, facilitating efficient downsampling while retaining essential spatial information. It involves dividing the point cloud into equally sized cubic voxel blocks with a side length of l. Overall, the process includes three steps:

1. Divide the point cloud into cubic voxel blocks with a side length of l, allowing for the calculation of the center coordinates of each voxel block, denoted by (x_c, y_c, z_c), as follows:

$$\begin{aligned} x_c &= l \cdot \left\lfloor \frac{x_i}{l} \right\rfloor + \frac{l}{2} \\ y_c &= l \cdot \left\lfloor \frac{y_i}{l} \right\rfloor + \frac{l}{2} \\ z_c &= l \cdot \left\lfloor \frac{z_i}{l} \right\rfloor + \frac{l}{2} \end{aligned} \tag{1}$$

2. For each voxel block, select a representative point based on a chosen strategy, such as the point closest to the center or the one closest to the plane.
3. Combine all the selected representative points to create a new point cloud, which represents the downsampled result.

The process of voxel downsampling is illustrated in Figure 4. By adjusting the side length (l) of the voxel blocks, we can control the density of the downsampled point cloud. A smaller l value retains more representative points and results in higher point cloud density, while a larger l value retains fewer representative points and leads to lower point cloud density. Therefore, in practical applications, the appropriate l value should be selected based on particular requirements.

Classical voxel filtering is a commonly used point cloud denoising method and is based on the idea of segmenting the point cloud into regular voxel grids and averaging the

points within each voxel. While this method effectively reduces point cloud noise, it has the following limitations:

- Loss of point cloud map information:Voxel filtering essentially involves the sampling and downsampling of point clouds. However, due to retaining only a single average value within each grid cell, it suffers from information loss. In applications where precise reconstruction and analysis of point cloud data are required, this information loss can lead to significant errors.
- Point cloud map blurring: Voxel filtering applies averaging operations to the points within each grid cell, potentially blurring fine details on the surfaces of objects in the point cloud map. This may pose issues in applications that require the preservation of detailed surface information.
- Inability to dynamically set downsampling ratios: Traditional point cloud downsampling methods, including classical voxel filtering, as well as other methods, such as grid-based downsampling, random downsampling, uniform downsampling, and clustering-based downsampling, cannot dynamically adjust downsampling ratios for point cloud maps with varying point counts and densities. Manual parameter tuning is required to set different downsampling factors for different point cloud maps, increasing human effort and parameter tuning time.

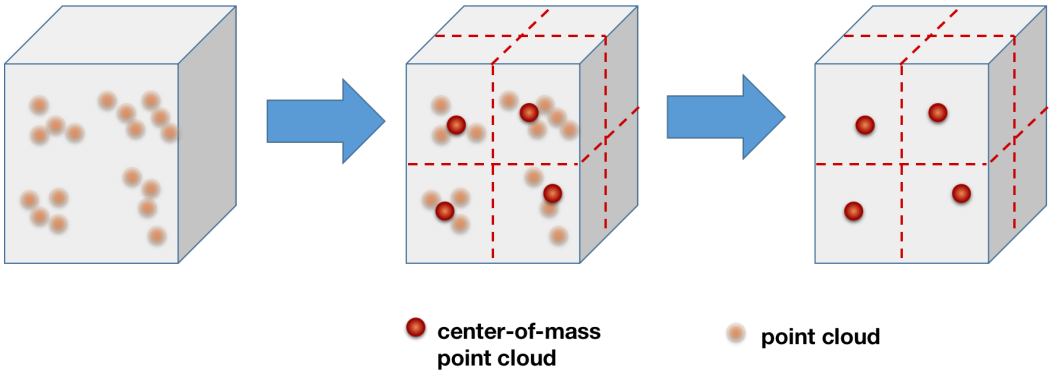


Figure 4. Voxel downsampling process schematic.

4. Methods

In response to the issues of information loss and blurring in classic voxel filtering for point cloud maps, we present an improved dynamic downsampling algorithm for 3D point cloud maps based on voxel filtering. The proposed method aims to dynamically segment 3D point cloud maps, preserving both edge information and crucial feature details. Moreover, this method adaptively segments voxels into various sizes, depending on the local density variations within the point cloud data, thereby reducing the need for manual intervention and parameter tuning.

The dynamic voxel-based 3D point cloud map downsampling algorithm consists of two modules: dynamic downsamplingand point cloud edge extraction. In the former, downsampling operations are tailored based on the density of points within each voxel block of the point cloud map. Specifically, if a unit voxel block has a higher density, it is subdivided into smaller voxel blocks, whereas if a unit voxel block has a lower density, it is subdivided into larger voxel blocks. This adaptive approach allows for different downsampling operations on various voxel blocks, addressing the drawback of classic voxel filtering, which rather tends to blur object characteristics by uniformly processing each voxel block. The point cloud edge extraction module determines whether points within the cloud represent an object's edge by analyzing the angle feature values of their normal vectors. Points identified as object edges are preserved and then merged with the downsampled point cloud, resulting in the final, combined point cloud map.

The flowchart of the dynamic voxel-based 3D point cloud map downsampling algorithm improved by voxel filtering is illustrated in Figure 5. Accordingly, the implementation steps of the improved algorithm are described below.

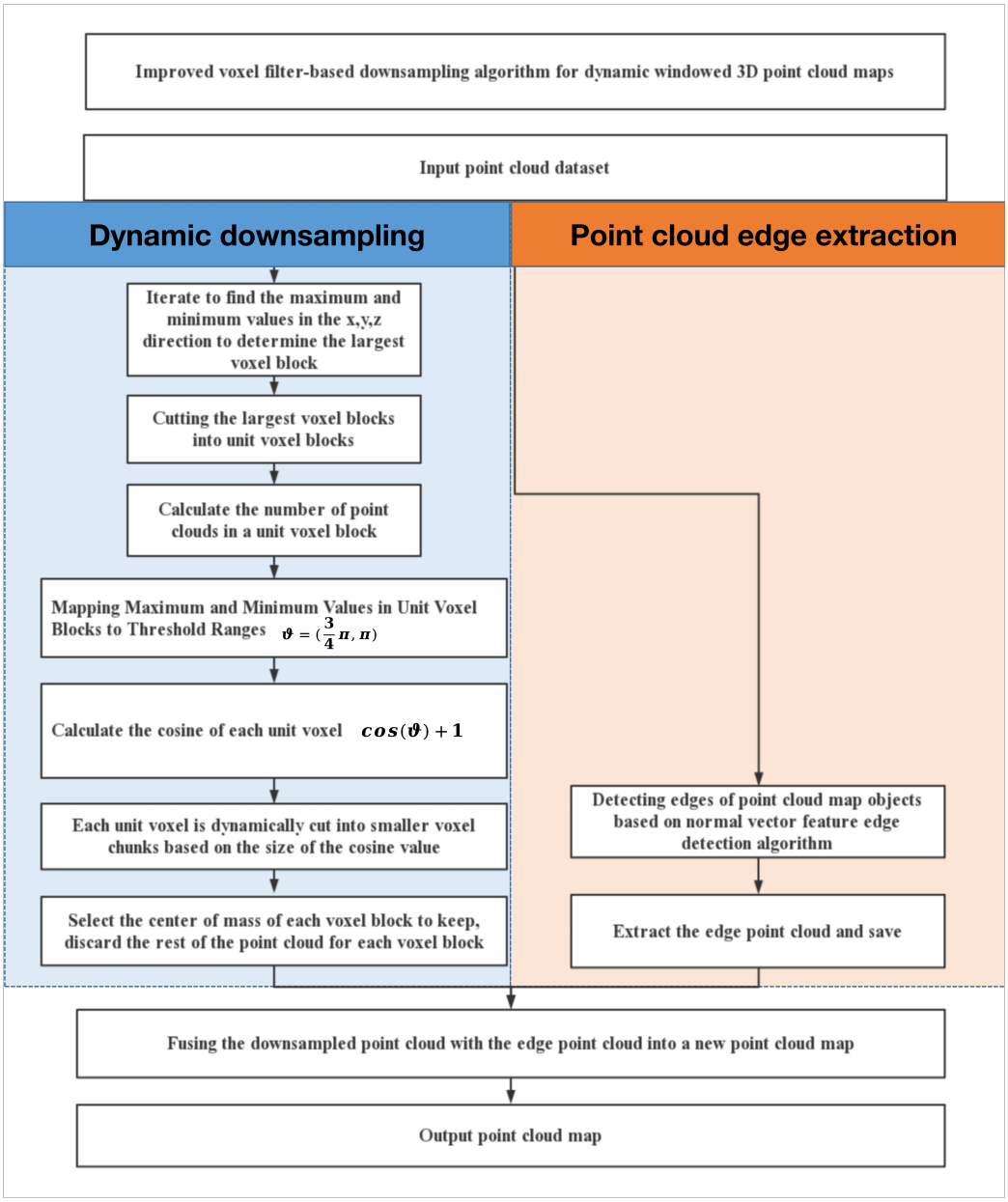


Figure 5. Flowchart of 3D point cloud map dynamic downsampling algorithm based on voxel filtering.

1. Input point cloud data: We begin with the input of 3D point cloud data.
2. Calculation of the maximum voxel block: The maximum values along the x-, y-, and z-axes of the point cloud data, denoted by X_{max}, Y_{max}, and Z_{max}, respectively, are determined. Additionally, the minimum values (X_{min}, Y_{min}, and Z_{min}) along these axes are also computed [34]. The edge length of the largest voxel block is determined by (2).

$$\begin{aligned} l_x &= X_{max} - X_{min} \\ l_y &= Y_{max} - Y_{min} \\ l_z &= Z_{max} - Z_{min} \end{aligned} \tag{2}$$

Thus, the largest voxel block is obtained, as shown in Figure 6.

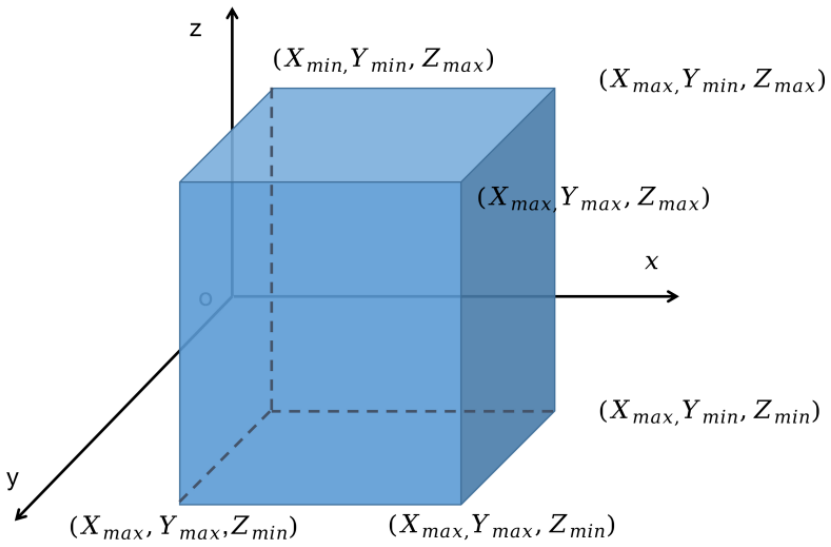


Figure 6. Maximum voxel block.

3. Voxel block division: The largest voxel block is divided into unit voxel blocks, with a designed edge length of L [35]. The division in three directions is described in (3).

$$\begin{aligned} M &= \left\lfloor \frac{l_x}{\text{cell}} \right\rfloor \\ N &= \left\lfloor \frac{l_y}{\text{cell}} \right\rfloor \\ L &= \left\lfloor \frac{l_z}{\text{cell}} \right\rfloor \end{aligned} \tag{3}$$

The summation of unit voxel blocks is illustrated in Figure 7.

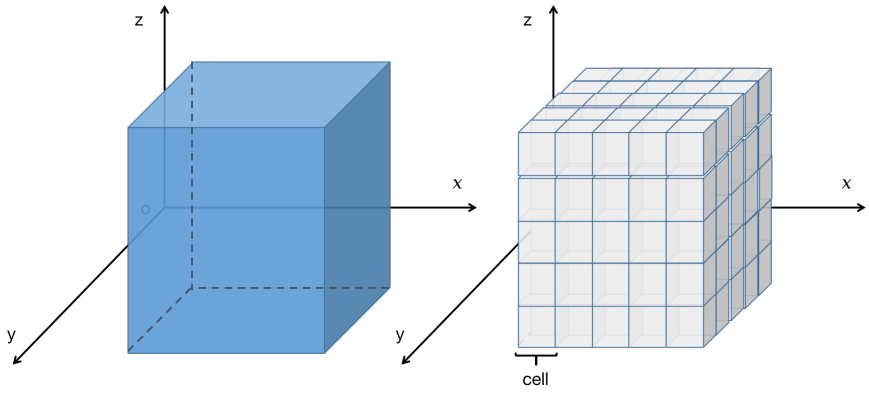


Figure 7. Subdivision of the Largest Voxel into Unit Voxels.

4. We count the number of points in each voxel block to obtain the point count collection for voxel blocks, denoted by $NUM = (n_1, n_2, n_3, \ldots, n_{SUM})$.
5. Subdivision of voxel blocks: By mapping $(\min(NUM), \max(NUM))$ to $\left(\frac{3}{4}\Pi, \Pi\right)$, we project the count of points within each unit voxel to the range of $\left(\frac{3}{4}\Pi, \Pi\right)$. This calculation involves subdividing voxel blocks based on the side length of each unit voxel. According to the property defined in (4) and (5), it is possible to subdivide voxel blocks into different sizes based on the quantity of points within each unit voxel. The cutting principle is illustrated in Figure 8.

$$\vartheta = \text{Normalize}\left(\text{NUM}, \left(\frac{3}{4}\Pi, \Pi\right)\right) \tag{4}$$

$$l_{\text{normalize}} = \cos(\vartheta) + 1 \tag{5}$$

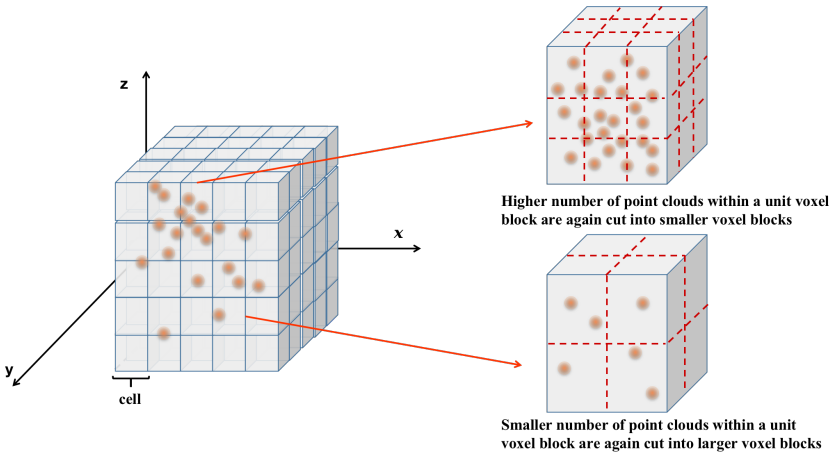


Figure 8. Dynamic subdivision of unit voxels.

6. We compute the centroid point within each voxel block, retain this centroid point, and discard the remaining points within the voxel block. We store the downsampled centroid point set. The principle is illustrated in Figure 9.

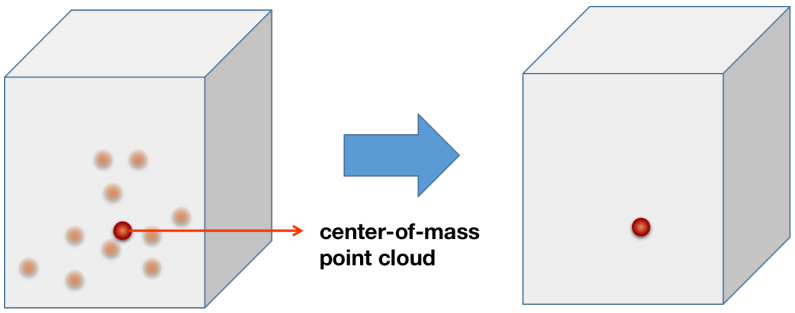


Figure 9. Preservation of centroid point cloud.

7. Detection of the characteristics of normal vectors to identify the edges of a point cloud. Let us assume a point cloud dataset with a point, denoted by p, having a normal vector N_p. The set of neighboring points around point p is denoted by $Q = q_1, q_2, \ldots, q_n$. The covariance matrix (C) can be computed as shown in (6).

$$C = \sum_{i=1}^{k} (q_i - \bar{p})(q_i - \bar{p})^T \tag{6}$$

where $\overline{\mathrm{p}} = \sum_{\mathrm{i}=1}^{\mathrm{k}} \frac{\mathrm{q_i}}{\mathrm{n}}$ represents the 3D point cloud coordinates of the neighboring point set around point p. The normal vector of point p is given by (7).

$$\mathrm{C} \cdot \mathrm{n_{p_j}} = \alpha_{\mathrm{j}} \cdot \mathrm{n_{p_i}} \tag{7}$$

where $\alpha_{\mathrm{j}} (j = 1, 2, 3)$ represents the eigenvalues of the covariance matrix (C).
The formula for calculating the feature quantity of the angle between normal vectors is given by (8). In flat regions, the angle between normal vectors is small, and in horizontal regions, the angle may even be 0, while in non-flat regions, the angle between normal vectors is relatively large.

$$n_p^k = \frac{1}{k}\sum_{i=1}^{k}\arccos\left(\frac{n_p \cdot n_{p^i}^k}{\|n_p\| \cdot \left\|n_p^k\right\|}\right) \tag{8}$$

8. The detected edge point set is merged with the downsampled centroid point set to create a new point set.

5. Experiments

5.1. Evaluation Metrics

The point cloud simplification rate is a well-known metric to measure the goodness of point cloud map downsampling. This simplification rate is calculated by (9).

$$R = \frac{\mathrm{NUM} - \mathrm{num}_d}{\mathrm{NUM}} \tag{9}$$

where NUM is the total number of points in the original point cloud, num_d is the total number of points in the downsampled point cloud, and R is the point cloud simplification ratio.

5.2. Experimental Results and Analysis

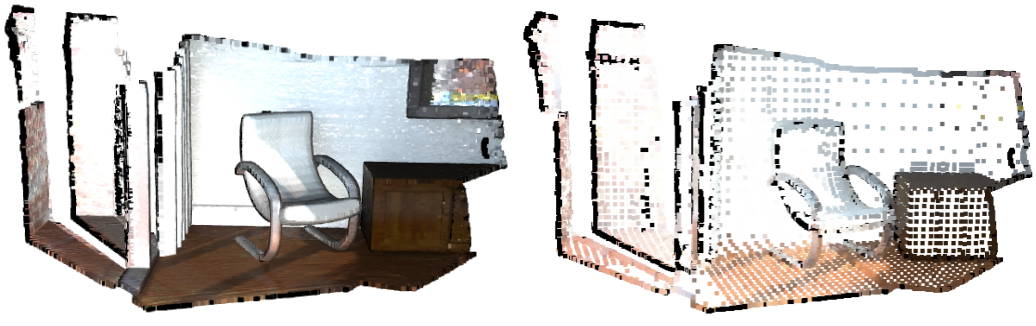
We used the nvida RTX3070 8G video memory in ubuntu 18.04, and the versions corresponding to the environment dependencies used are shown in Table 2.

Table 2. List of environmental dependencies.

System Environment	Version
cuda	11.0
conda	23.1.0
python	3.7
pytorch	1.8.1

Our algorithm was evaluated against traditional downsampling techniques, namely, voxel downsampling, uniform downsampling, random downsampling, grid downsampling, clustering-based downsampling, and farthest-point sampling (FPS), through comparative experiments. The original point cloud used in these experiments consisted of 196,133 points [36].

As shown in Figure 10, the point cloud processed by our method (Figure 10b) has significantly reduced data volume while preserving key boundary information compared to the original point cloud (Figure 10a).



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

Figure 10. Comparison between original and proposed methods. Images generated with (**a**) original method and (**b**) our method.

Our algorithm is capable of identifying and prioritizing the retention of key points that define the shape and structure of objects.

Figure 11a–c demonstrate that voxel downsampling mandates the manual tuning of the leafsize parameter. A large leafsize overly sparsens the cloud, erasing critical features, whereas a small leafsize inadequately simplifies the cloud, failing to efficiently reduce data complexity.

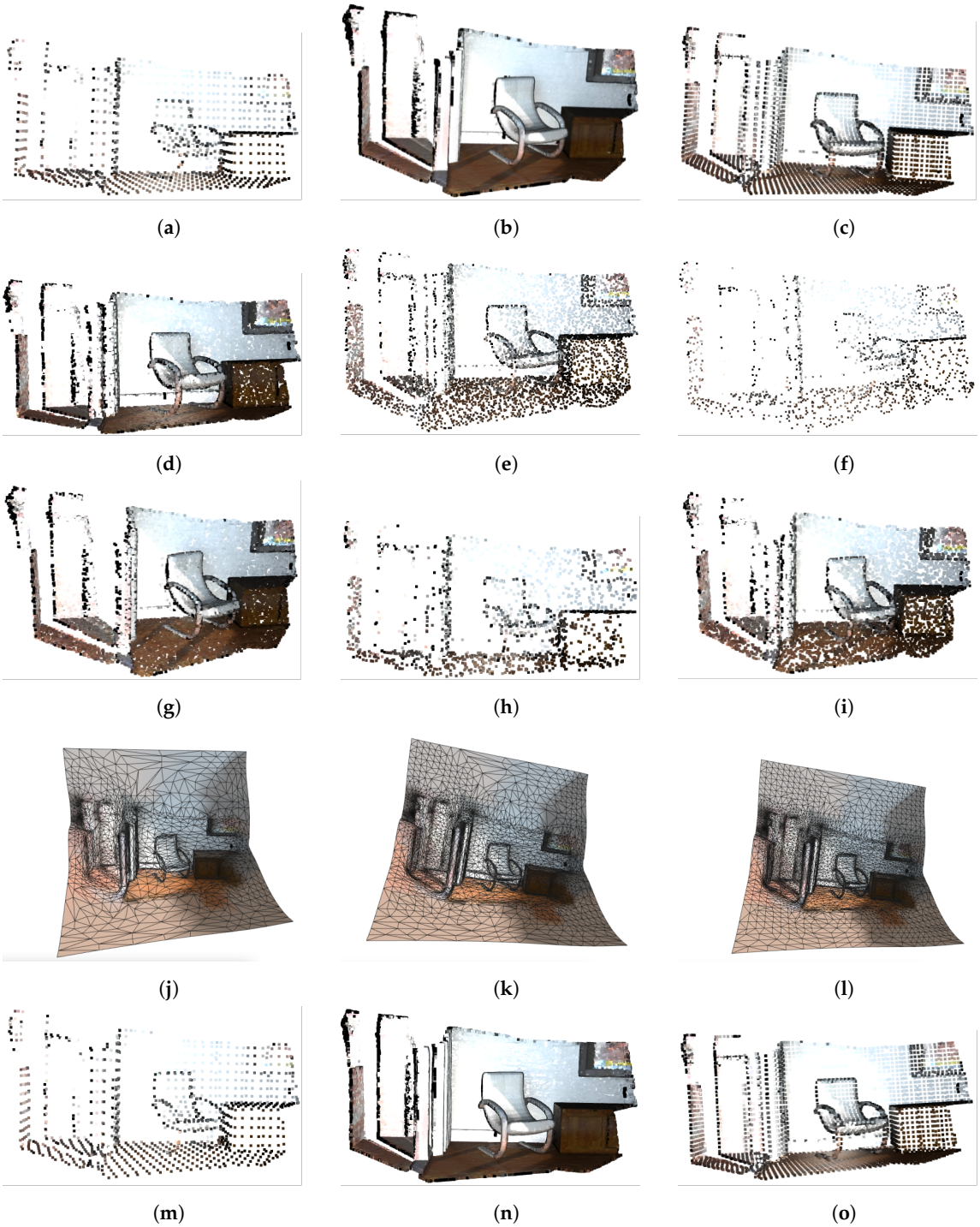
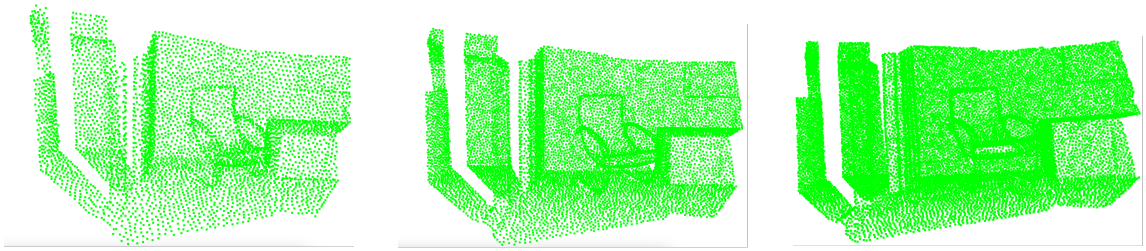


Figure 11. *Cont.*



(**p**) (**q**) (**r**)

Figure 11. Comparative experimental evaluation of various downsampling methods. Voxel downsampling: (**a**) leafsize = 0.1, (**b**) leafsize = 0.01, and (**c**) leafsize = 0.05. Uniform downsampling: (**d**) every k points = 10, (**e**) every k points = 25, and (**f**) every k points = 100. Random downsampling: (**g**) leafsize = 0.1, (**h**) leafsize = 0.01, and (**i**) leafsize = 0.05. Grid downsampling: (**j**) target number of triangles = 5000, (**k**) target number of triangles = 10,000, and (**l**) target number of triangles = 15,000. Cluster downsampling (**m**) leafsize = 0.1, (**n**) leafsize = 0.01, and (**o**) leafsize = 0.05. FPS: (**p**) target number of triangles = 5000, (**q**) target number of triangles = 10,000, and (**r**) target number of triangles = 15,000.

Figure 11d–f reveal that uniform downsampling in the PCL library also necessitates manual adjustment, this time of the every_k_points parameter. Higher values of every_k_points lead to excessive data loss, while lower values do not simplify the cloud enough. Our proposed method eliminates the need for such adjustments, automatically adapting to cloud scale and density, thereby preserving edge and feature integrity more effectively.

As shown in Figure 11g–i, the proposed downsampling algorithm requires artificial adjustment of the leafsize coefficient, like random downsampling in the PCL library. The difference from voxel downsampling is that if leafsize is too small, the sampling is sparser, and the original points are lost. For cloud feature information, the larger the leafsize, the denser the sampling. Compared with the random downsampling method in the PCL library, the proposed algorithm saves the cost of manual parameter adjustment. It can dynamically segment point cloud maps according to different scales of point cloud maps and different numbers of local point clouds and can effectively preserve the edge of the point cloud image and the feature information of the object.

Figure 11j–l illustrate the necessity of manually setting target number of triangles for grid downsampling. This parameter directly influences the downsampling density, with lower values resulting in significant data loss and higher values inadequately simplifying the cloud. Unlike this method, our algorithm requires no manual parameter adjustment and dynamically adjusts parameters to the cloud's complexity, ensuring optimal simplification while retaining critical structural details.

Figure 11m–o illustrate that clustering downsampling, like voxel downsampling, necessitates the manual tuning of the leafsize parameter. Oversized leafsize values lead to excessive sparsity and feature loss, whereas too small values result in insufficient simplification, failing to meet the goals of effective point cloud reduction.

The FPS (farthest-point sampling) method demands manual specification of the output size, denoted by object number of triangles. A lower count results in overly sparse clouds and potential feature loss, while a higher count compromises the simplification objective. Figure 11p–r display the outcomes of varying the FPS parameters.

Table 3 provides a comparative analysis showcasing the effectiveness of our dynamic downsampling algorithm against traditional methods. Notably, it achieves a superior balance between simplification rate (91.89%) and processing speed (0.01289 s), highlighting its efficiency and practicality for real-time applications. The algorithm significantly reduces the data volume to 15,906 downsampled points from the original 196,133, maintaining essential details without the extensive manual parameter tuning required by other methods. This adaptability not only streamlines the downsampling process but also enhances usability,

setting a new benchmark in 3D point cloud processing. The comparison underscores the algorithm's potential to revolutionize point cloud processing tasks, offering a user-friendly, efficient, and effective solution for various applications.

Table 3. Experimental comparison table of various downsampling methods.

Methods	Number of Downsampled Points	Simplification Rate	Time (s)	Parameters
Voxel downsampling	94,743	0.5169	0.1025	leafsize = 0.01
	4718	0.9759	0.0125	leafsize = 0.05
	1284	0.9934	0.0085	leafsize = 0.1
Random downsampling	1961	0.9900	0.0143	leafsize = 0.01
	9806	0.9500	0.0126	leafsize = 0.05
	19613	0.9000	0.0135	leafsize = 0.1
Uniform downsampling	1962	0.9899	0.1129	every_k_points = 100
	7846	0.9599	0.3493	every_k_points = 25
	19,614	0.9000	1.0615	every_k_points = 10
Grid downsampling	5000	0.9745	3.1347	arget_number_of _triangles = 5000
	10,000	0.9490	3.5045	arget_number_of _triangles = 10,000
	15,000	0.9235	4.0243	arget_number_of _triangles = 15,000
Cluster downsampling	95,240	0.5144	1.0808	leafsize = 0.01
	4830	0.9753	1.007	leafsize = 0.05
	1248	0.9936	0.9958	leafsize = 0.1
FPS	1000	0.9745	34.8717	arget_number_of _triangles = 5000
	5000	0.9490	65.9702	arget_number_of _triangles = 10,000
	10,000	0.9235	109.9000	arget_number_of _triangles = 15,000
Our method	15,906	0.9189	0.0129	

6. Discussion

The findings presented in this study underscore the efficacy of the proposed dynamic downsampling algorithm for 3D point cloud maps, particularly in terms of simplification rate, processing speed, and the reduced necessity for manual parameter adjustment. This discussion further explores these results in the context of existing research, practical applications, and future directions.

6.1. Comparison with Existing Methods

Our dynamic downsampling algorithm demonstrates significant improvements over traditional downsampling techniques, such as voxel, random, uniform, grid, and cluster downsampling, as well as farthest-point sampling (FPS). The key advantages include a balance between high simplification rates and rapid processing times, crucial for real-time processing applications. Unlike previous methods that often require labor-intensive parameter tuning to optimize performance, our algorithm's adaptive nature significantly reduces this burden, offering a more user-friendly approach. These enhancements align with the growing demand for efficient point cloud processing in applications like autonomous driving, robotics, and smart city planning.

6.2. Practical Implications

The practicality of our algorithm extends beyond its immediate performance benefits. By facilitating a faster and more intuitive downsampling of 3D point cloud data, the algorithm enables more efficient subsequent processing tasks, such as segmentation and classification. This efficiency can profoundly impact industries relying on 3D scanning and modeling technologies, where processing speed and data quality directly influence operational effectiveness and innovation capabilities.

6.3. Limitations and Future Work

While our algorithm represents a substantial advancement in point cloud downsampling, certain limitations warrant further investigation. For instance, the algorithm's performance in extremely dense or noisy environments remains an area for improvement. Future research could explore enhancements to the edge preservation mechanism or the introduction of noise-resistant features to address these challenges.

Moreover, the adaptability of the algorithm to various point cloud data types and sources could be further examined. Extensive testing across a broader spectrum of datasets will help refine the algorithm's applicability and efficiency across different scenarios.

7. Conclusions

This study introduced a novel dynamic downsampling algorithm for 3D point cloud maps, showcasing its superiority in balancing simplification rates and processing speeds while minimizing the need for manual parameter tuning. Our contributions, through the development and validation of this algorithm, address critical challenges in the processing of large-scale 3D point cloud data, offering significant advancements over traditional downsampling methods.

The implementation of our algorithm demonstrates not just a technical achievement, but also promises substantial practical benefits across various applications. From autonomous driving systems and robotics to augmented reality and urban planning, the efficient processing of 3D point cloud data is foundational. Our work paves the way for more streamlined data analysis processes, potentially unlocking new innovations and improvements in these fields.

However, the journey to perfecting downsampling algorithms is far from complete. Despite the promising results, our algorithm, like all methods, has its limitations. The performance in scenarios with extreme density variations and noise levels presents an opportunity for further research. Additionally, the exploration of the algorithm's adaptability across different types of point cloud data sources could yield even more versatile and robust solutions.

Looking ahead, we anticipate a multifaceted approach to future research. Efforts will likely focus on enhancing the algorithm's ability to preserve finer details in highly complex environments, improve noise resilience, and further reduce computational demands. Moreover, integrating machine learning techniques to dynamically adjust downsampling parameters based on the specific characteristics of each point cloud dataset could offer a pathway to even more sophisticated and automated processing tools.

In conclusion, our dynamic downsampling algorithm represents a significant step forward in the field of 3D point cloud processing. It addresses several longstanding challenges and opens up new possibilities for both academic research and practical applications. We remain committed to advancing this field, inspired by the potential to contribute to the next generation of technologies that rely on 3D point cloud data.

Author Contributions: Conceptualization, W.L.; methodology, W.L.; ;validation, H.Z.; formal analysis, W.L. and X.M.; investigation, W.K. and H.S.; resources, W.K.; data curation, H.Z.; writing—original draft preparation, W.L.; writing—review and editing, W.K. and H.S.; visualization, H.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key R&D Program of China, grant number 2022YFC3803600; the National Natural Science Foundation of China, grant number 62372023; and the Science and Technology Development Fund, Macau SAR, file number 0122/2023/AMJ.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are contained within the article.

Acknowledgments: This study was partially supported by the National Key R&D Program of China (No. 2022YFC3803600), the National Natural Science Foundation of China (No. 62372023), the Science and Technology Development Fund, Macau SAR (file No. 0122/2023/AMJ), and the Open Fund of the State Key Laboratory of Software Development Environment (No. SKLSDE-2023ZX-11). The authors appreciate the support from HAWKEYE Group.

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

References

1. Rozenberszki, D.; Majdik, A.L. LOL: Lidar-only odometry and localization in 3D point cloud maps. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE, Virtual, 31 May–31 August 2020; pp. 4379–4385.
2. Cui, Y.; Chen, R.; Chu, W.; Chen, L.; Tian, D.; Li, Y.; Cao, D. Deep learning for image and point cloud fusion in autonomous driving: A review. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 722–739. [CrossRef]
3. Fernandes, D.; Silva, A.; Névoa, R.; Simões, C.; Gonzalez, D.; Guevara, M.; Novais, P.; Monteiro, J.; Melo-Pinto, P. Point-cloud based 3D object detection and classification methods for self-driving applications: A survey and taxonomy. *Inf. Fusion* **2021**, *68*, 161–191. [CrossRef]
4. Trapp, M.; Dumke, F.; Döllner, J. Occlusion management techniques for the visualization of transportation networks in virtual 3D city models. In Proceedings of the 12th International Symposium on Visual Information Communication and Interaction, Shanghai, China, 20–22 September 2019; pp. 1–8.
5. Wang, S.; Yang, D.; Sheng, H.; Shen, J.; Zhang, Y.; Ke, W. A Blockchain-enabled Distributed System for Trustworthy and Collaborative Intelligent Vehicle Re-identification. *IEEE Trans. Intell. Veh.* **2023**. [CrossRef]
6. Yang, L.; Liu, Y.; Peng, J.; Liang, Z. A novel system for off-line 3D seam extraction and path planning based on point cloud segmentation for arc welding robot. *Robot. Comput. Integr. Manuf.* **2020**, *64*, 101929. [CrossRef]
7. Wegen, O.; Döllner, J.; Wagner, R.; Limberger, D.; Richter, R.; Trapp, M. Non-Photorealistic Rendering of 3D Point Clouds for Cartographic Visualization. *Abstr. ICA* **2022**, *5*, 1–2. [CrossRef]
8. Sheng, H.; Zhang, Y.; Wang, W.; Shan, Z.; Fang, Y.; Lyu, W.; Xiong, Z. High confident evaluation for smart city services. *Front. Environ. Sci.* **2022**, *10*, 950055. [CrossRef]
9. Klimke, J. Web-Based Provisioning and Application of Large-Scale Virtual 3D City Models. Ph.D. Thesis, Universität Potsdam, Potsdam, Germany, 2018. [CrossRef]
10. Verajagadheswa, P.; Kandasamy, P.S.; Elangovan, K.; Elara, M.R.; Bui, M.V.; Le, A.V. A novel autonomous staircase cleaning system with robust 3D-Deep Learning-based perception technique for Area-Coverage. *Expert Syst. Appl.* **2022**, *194*, 116528. [CrossRef]
11. Zhou, Y.; Tuzel, O. Voxelnet: End-to-end learning for point cloud based 3D object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4490–4499.
12. Chen, H.; Xie, T.; Liang, M.; Liu, W.; Liu, P.X. A local tangent plane distance-based approach to 3D point cloud segmentation via clustering. *Pattern Recog.* **2023**, *137*, 109307. [CrossRef]
13. Zhang, H.; Wang, C.; Tian, S.; Lu, B.; Zhang, L.; Ning, X.; Bai, X. Deep learning-based 3D point cloud classification: A systematic survey and outlook. *Displays* **2023**, *79*, 102456. [CrossRef]
14. Li, J.; Saydam, S.; Xu, Y.; Liu, B.; Li, B.; Lin, X.; Zhang, W. Class-aware tiny object recognition over large-scale 3D point clouds. *Neurocomputing* **2023**, *529*, 166–181. [CrossRef]
15. Sheng, H.; Zhang, Y.; Chen, J.; Xiong, Z.; Zhang, J. Heterogeneous association graph fusion for target association in multiple object tracking. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *29*, 3269–3280. [CrossRef]
16. Sheng, H.; Zhao, P.; Zhang, S.; Zhang, J.; Yang, D. Occlusion-aware depth estimation for light field using multi-orientation EPIs. *Pattern Recognit.* **2018**, *74*, 587–599. [CrossRef]
17. Sheng, H.; Liu, X.; Zhang, S. Saliency analysis based on depth contrast increased. In Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, Shanghai, China, 20–25 March 2016; pp. 1347–1351.
18. Yu, C.; Lei, J.; Peng, B.; Shen, H.; Huang, Q. SIEV-Net: A structure-information enhanced voxel network for 3D object detection from LiDAR point clouds. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5703711. [CrossRef]
19. He, Q.; Wang, Z.; Zeng, H.; Zeng, Y.; Liu, Y. Svga-net: Sparse voxel-graph attention network for 3d object detection from point clouds. In Proceedings of the AAAI Conference on Artificial Intelligence, Philadelphia, PA, USA, 27 February–2 March 2022; Volume 36, pp. 870–878.

20. Que, Z.; Lu, G.; Xu, D. Voxelcontext-net: An octree based framework for point cloud compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 6042–6051.
21. Alexa, M.; Behr, J.; Cohen-Or, D.; Fleishman, S.; Levin, D.; Silva, C.T. Computing and rendering point set surfaces. *IEEE Trans. Vis. Comput. Graph.* **2003**, *9*, 3–15. [CrossRef]
22. Gelfand, N.; Ikemoto, L.; Rusinkiewicz, S.; Levoy, M. Geometrically stable sampling for the ICP algorithm. In *Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling, Banff, AL, Canada, 6–10 October 2003*; 3DIM 2003, Proceedings; IEEE: Piscataway, NJ, USA, 2003; pp. 260–267.
23. Xuepeng, C. An Under-sampling Algorithm Based on K-means Clustering. *Bull. Sci. Technol.* **2013**, *29*, 73–75.
24. Xiao, Z.; Gao Jian, W.D.; Lanyu, Z. Voxel Grid Downsampling for 3D Point Cloud Recognition. *Modul. Mach. Tool Autom. Manuf. Tech.* **2022**, *11*, 43–47.
25. Nguyen, D.T.; Quach, M.; Valenzise, G.; Duhamel, P. Learning-based lossless compression of 3D point cloud geometry. In Proceedings of the ICASSP 2021–2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, Toronto, ON, Canada, 6–11 June 2021; pp. 4220–4224.
26. Qin, P.; Zhang, C.; Dang, M. GVnet: Gaussian model with voxel-based 3D detection network for autonomous driving. *Neural Comput. Appl.* **2022**, *34*, 6637–6645. [CrossRef]
27. Yang, Y.; Feng, C.; Shen, Y.; Tian, D. Foldingnet: Point cloud auto-encoder via deep grid deformation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 206–215.
28. Gezawa, A.S.; Bello, Z.A.; Wang, Q.; Yunqi, L. A voxelized point clouds representation for object classification and segmentation on 3D data. *J. Supercomput.* **2022**, *78*, 1479–1500. [CrossRef]
29. Garland, M.; Willmott, A.; Heckbert, P.S. Hierarchical face clustering on polygonal surfaces. In Proceedings of the 2001 Symposium on Interactive 3D Graphics, Chapel Hill, NC, USA, 26–29 March 2001; pp. 49–58.
30. YUAN Hua, P.J.; Jianwen, M. Research on Simplification Algorithm of Point Cloud Based on Voxel Grid. *Video Eng.* **2015**, *39*, 43–47.
31. Zhang Liyan, N.J.Z.; Zhou, L. Research on Adaptive Remeshing of Triangle Meshes. *J. Comput. Aided Des. Comput. Graph.* **2002**, *14*, 204–208.
32. Rusinkiewicz, S.; Levoy, M. Efficient variants of the ICP algorithm. In Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling, IEEE, Quebec City, QC, Canada, 28 May–1 June 2001; pp. 145–152.
33. Zhang, P.; Xiao, Y.; Wang, X.; Duan, B. Semantic segmentation of point clouds of field obstacle-crossing terrain for multi-legged rescue equipment based on random forest. In Proceedings of the 2020 International Conference on Artificial Intelligence and Electromechanical Automation (AIEA), IEEE, Tianjin, China, 26–28 June 2020; pp. 147–153.
34. Chen, Z.; Li, L.; Niu, K.; Wu, Y.; Hua, B. Pose measurement of non-cooperative spacecraft based on point cloud. In Proceedings of the 2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC), IEEE, Xiamen, China, 10–12 August 2018; pp. 1–6.
35. Yang, J.; Wang, C.; Luo, W.; Zhang, Y.; Chang, B.; Wu, M. Research on point cloud registering method of tunneling roadway based on 3D NDT-ICP algorithm. *Sensors* **2021**, *21*, 4448. [CrossRef] [PubMed]
36. Zhou, Q.Y.; Park, J.; Koltun, V. Open3D: A modern library for 3D data processing. *arXiv* **2018**, arXiv:1801.09847.