


Article

A Pre-Precession Module for Point-Based Deep Learning in Dense Point Clouds in the Ship Engineering Field

Shilin Huo ¹, Yujun Liu ¹, Ji Wang ^{1,2,3,*} , Rui Li ¹, Xiao Liu ¹ and Jiawei Shi ¹

¹ School of Naval Architecture and Ocean Engineering, Dalian University of Technology, Dalian 116024, China; huoshilin@mail.dlut.edu.cn (S.H.); yjliu@dlut.edu.cn (Y.L.); lirui@dlut.edu.cn (R.L.); liuxiao@dlut.edu.cn (X.L.); sjw123@mail.dlut.edu.cn (J.S.)

² Collaborative Innovation Center for Advanced Ship and Deep-Sea Exploration, Shanghai 200240, China

³ State Key Laboratory of Structural Analysis for Industrial Equipment, Dalian 116024, China

* Correspondence: wangji@dlut.edu.cn; Tel.: +86-188-4090-2302; Fax: +86-0411-84706506

Abstract: Recently, point cloud technology has been applied in the ship engineering field. However, the dense point cloud acquired by terrestrial laser scanning (TLS) technology in ship engineering applications brings an obstacle to some powerful and advanced point-based deep learning point cloud processing methods. This paper presents a deep learning pre-precession module to ensure the feasibility of processing dense point clouds on commonly available computer devices. The pre-precession module is designed according to the traditional point cloud processing methods and the PointNet++ paradigm, and is evaluated on two ship structure datasets and two popular point cloud datasets. Experimental results illustrate that (i) the proposed module improves the performance of point-based deep learning semantic segmentation networks, and (ii) the proposed module empowers the existing point-based deep learning networks with the capability to process dense input point clouds. The proposed module may provide a useful semantic segmentation tool for realistic dense point clouds in various industrial applications.

Keywords: intelligent manufacturing; ship structure; laser scanning; deep learning; point cloud



Citation: Huo, S.; Liu, Y.; Wang, J.; Li, R.; Liu, X.; Shi, J. A Pre-Precession Module for Point-Based Deep Learning in Dense Point Clouds in the Ship Engineering Field. *J. Mar. Sci. Eng.* **2023**, *11*, 2248. <https://doi.org/10.3390/jmse11122248>

Academic Editor: Marco Cococcioni

Received: 12 October 2023

Revised: 23 November 2023

Accepted: 23 November 2023

Published: 28 November 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Recently, advanced point cloud technology has been applied in the ship engineering field. For instance, in the construction quality analysis of hull blocks [1], steel parts reconstruction [2], ship overload identification [3], ship berthing angle estimation [4], and 3D target detection for ship navigation [5]. However, the dense point cloud acquired by terrestrial laser scans (TLS) technology in the ship engineering field (Figure 1 shows two instances) presents an obstacle to point cloud processing technologies. In particular, some powerful and advanced deep learning methods struggle to handle dense point sets. This paper aims to develop a feasible method to ensure the precession of dense point clouds in deep learning applications.

Since point-based neural networks directly process raw point data and avoid the quantization of artifacts [6] generated by the 3D grid, or voxel-based approaches [7], they are indisputably the most advanced point cloud deep learning methods. All of the point-based neural networks are inspired by the milestone work of PointNet [6], who designed a sophisticated network structure to achieve permutation invariance of input points. The most significant successors are PointNet++ [8], who considered the local features of the whole point cloud, and the Dynamic Graph Convolution Neural Network (DGCNN) [9], who applied the graph neural network to generate edge features in the feature spaces. Extensive variants of these three methods have been developed for various applications. In the application of autonomous driving and augmented reality, 3D Single-Stage Detectors (3DSSD) [10] utilized the PointNet++ backbone and considered furthest-point-sampling

(FPS) in the feature spaces to predict the 3D box containing classification objects. Instance-Aware Single-Stage Detectors (IA-SSD) [11] also used the Set Abstraction (SA) layers in PointNet++ to aggregate local features. The Point-Voxel Region-based Convolution Neural Network (PV-RCNN++) [12] combined the 3D CNN and PointNet++ to predict the RoI-grid. In the 3D registration application, Deep Closest Point (DCP) [13], the Inliers Estimation Network (INENet) [14], and You Only Hypothesize Once (YOHO) [15] applied the PointNet++ modules to generate the features of all points, and then predicted point-corresponding relationships according to the feature distances. The PointNet branch plays an important role in the feature aggregation step of modern 3D registration deep learning methods. In the 3D scene segmentation application, the Linked Dynamic Graph Convolution Neural Network (LDGCNN) [16] and the Dual-Graph Attention Convolution Network (DGACN) [17] extended the DGCNN method to a linked feature version and a dual graph version, respectively. PatchFormer [18] developed an efficient point transformer according to the PointNet modules and the transformer paradigm [19]. In recent years, point-based neural networks have become a pivotal component in the point cloud deep learning field, and are developing rapidly.

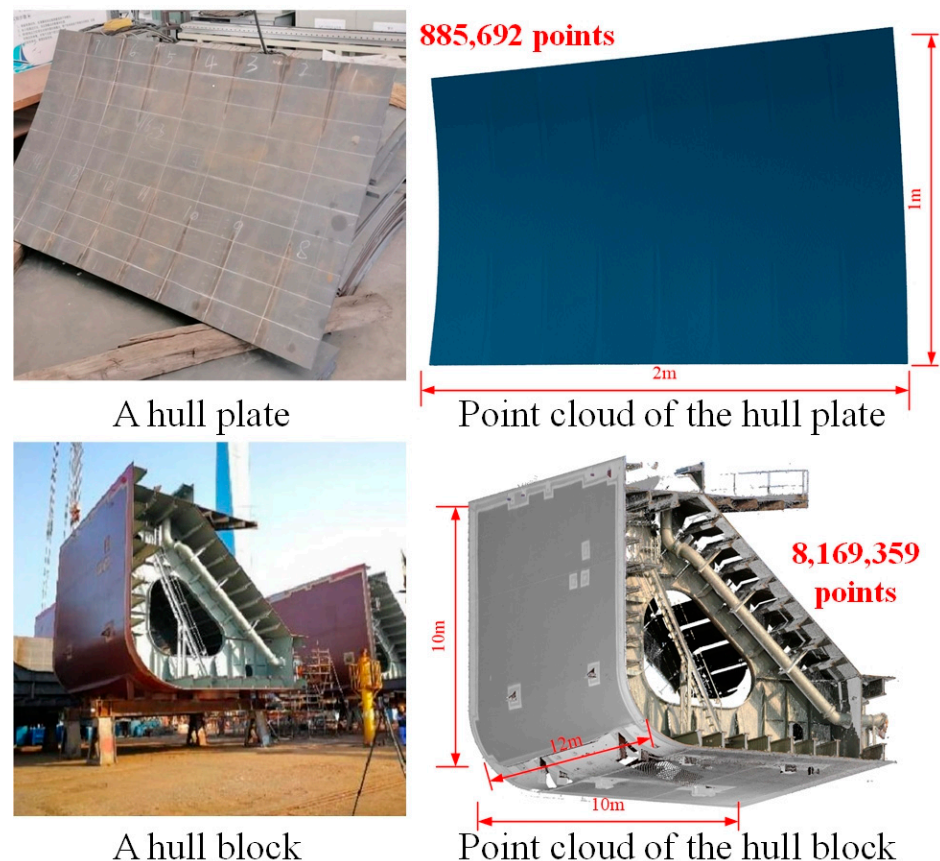


Figure 1. Two dense point cloud instances. The top row shows the scanning point cloud of a hull plate containing 885,692 points, and the bottom row shows the scanning point cloud of a hull block consisting 8,169,359 points.

However, most point-based neural networks are designed for experimental datasets (such as the well-known datasets ModelNet [20], KITTI [21], nuScenes [22], and 3DMatch [23]), and these methods typically involve a relatively small number of sampled points (usually 2048, 4096 or 8192 points) during the processing phase. This point scale clearly differs significantly from the realistic point cloud in the ship engineering field. In some tasks, such as classification, a small number of sampled points do not significantly impact the overall performance of deep learning networks. However, in other applications, such as semantic segmentation, sparse sampling disregards local information in the original data, resulting

in a decrease in resolution and making it difficult to obtain detailed processing results of the whole point cloud. One possible processing approach is to manually design a method after obtaining semantic segmentation results through point-based neural networks, which uses the prediction results of the sampled points to infer the labels of the remaining points in the original point cloud; however, this approach clearly lacks accuracy and persuasiveness. Obviously, developing a network that can directly achieve semantic segmentation from a dense input point cloud is evidently more reasonable. Furthermore, in situations where high-precision prediction or regression results are required, using a large number of raw points can fully take into account the information in the original scanning data, which may better improve the prediction or regression results of deep learning methods.

Therefore, for point clouds in the ship engineering field, a point-based deep learning structure that can handle dense input points is needed. Most point-based deep learning methods, when dealing with dense input points, result in memory and time consumption that is unbearable for general researchers. Based on traditional point cloud processing methods and the PointNet++ paradigm, we propose a pre-processing module which allows for the calculation of dense point sets in common commercial graphics cards. Employing the traditional point segmentation methods, the whole input point cloud is divided into multiple sub-regions, and based on the performance of the available graphics card, the input point cloud is processed in a time-sharing and batch-wise manner. Using a modified PointNet++ module, these sub-regions are reorganized to achieve successful execution of dense point sets on common GPUs. The proposed module may provide a powerful tool for realistic point clouds in various industrial applications.

The remainder of this paper is organized as follows. Section 2 introduces the proposed pre-processing module for a point-based deep learning method. To evaluate the performance of the proposed pre-processing module, it is integrated into some existing PointNet variants, and the experiment results are shown in Section 3. Finally, conclusions are drawn in Section 4.

2. The Proposed Pre-Processing Module

The proposed pre-processing module consists of two parts, a pre-segmentation part and a modified PointNet++ module (as shown in Figure 2). In the first part, traditional point segmentation methods are employed to divide the dense input point sets into smaller regions. In the second part, users can determine the number of regions processed in each iterative step based on the available GPU memory, enabling the computation of the dense point cloud in multiple steps. The proposed module is a convenient and efficient method with low device requirements. It can be flexibly integrated with any PointNet++ variant, and Section 2.2 illustrates the process of combining it with conventional point-based neural networks.

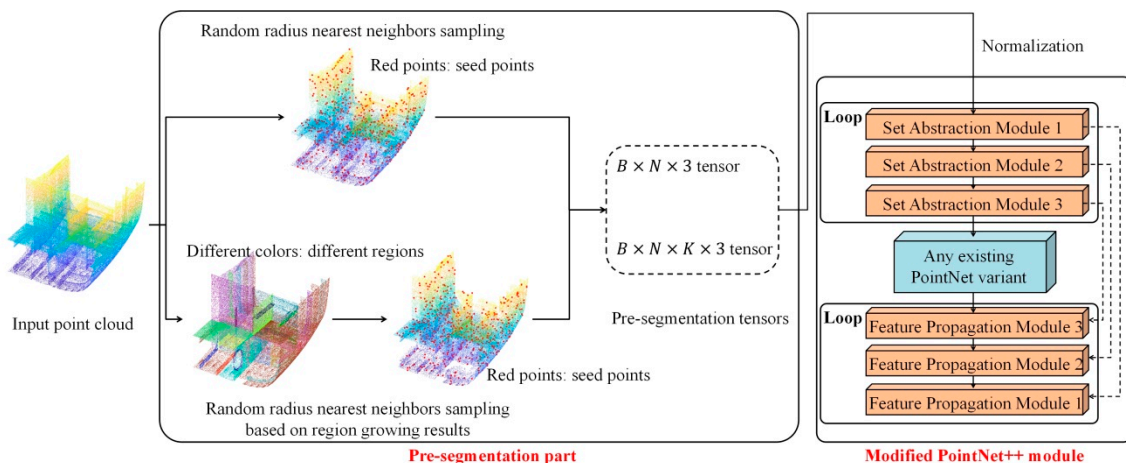


Figure 2. The framework of the proposed pre-processing module.

2.1. Pre-Segmentation Methods

First, we assume that the point number of the input point cloud is M , which will be divided into N regions, with each region containing K points. Note that there is no requirement for M to be equal to $N \times K$. This is because it is generally difficult to divide a point cloud into N equal parts, and an extreme instance is that M is a prime number. Therefore, this paper only requires that these N regions contain a sufficient number of total points.

The most simplistic method of pre-segmentation is to haphazardly select N subsets from the original data, each containing K individuals (as shown in Figure 3). The members within these subsets have no relationship or connection with each other, as the selection process is done without any conscious thought or consideration. In PointNet, if the consideration of the T-Net is disregarded, this approach is feasible since the map constructed by the multi-layer perceptron (MLP) ignores the interconnection among the points before the max pool step. However, more advanced point-based methods incorporate the PointNet++ structure to generate CNN-like hierarchical features that reflect local geometric properties, and this simple segmentation approach appears to be unreasonable.

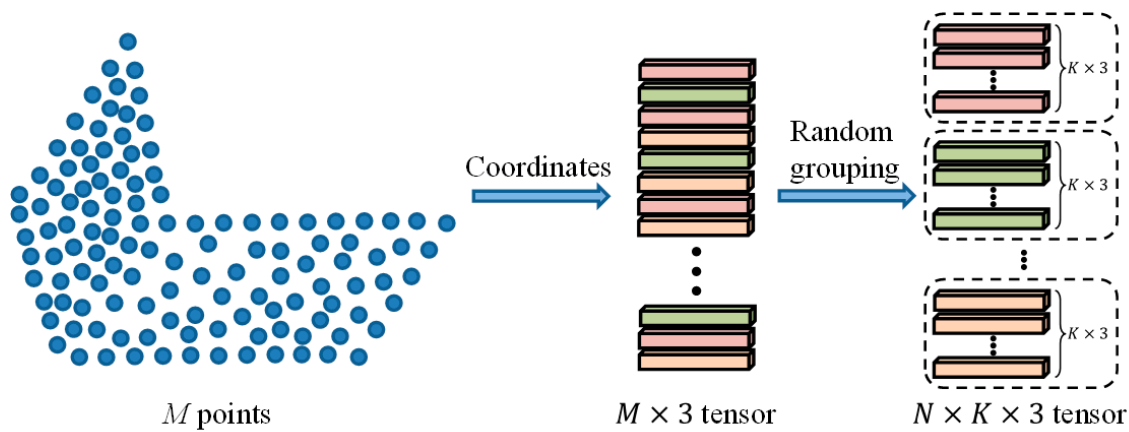


Figure 3. Haphazard selection of the original data. Each elongated, colored cube represents a 1×3 tensor.

We present two segmentation methods here: random radius nearest neighbors (RNN) sampling and random RNN sampling based on region growing results.

RNN is a data search approach based on the kd-tree [24] data structure. It can be implemented through the K-Nearest Neighbors (KNN) method [25] in some simpler applications. It is an important technique in the computer graphics field. Our random RNN sampling method begins by selecting N seed points from the whole M points. Subsequently, for each seed point, we search for all of the nearest neighbors of the query point in a given radius R . If the count of these neighbors is greater than or equal to K , we randomly select K members from these neighbors. If the count is less than K , we augment these neighbors by adding some duplicated instances of that seed point to ensure that the neighbors contain K members (as shown in Figure 4). Figure 5 illustrates the workflow of the random RNN sampling method.

Our random RNN sampling based on region growing results contains two steps: region growing of the original point cloud and random RNN sampling of each region. Region growing [26] is a popular traditional method for point cloud segmentation, which exploits curvatures [27] and the angles of normal vectors between neighboring points to achieve growth and expansion of the initial random seed points. It divides the original point cloud into simple sub-regions with coherent geometric structures. After obtaining the sub-regions of the original point cloud, each sub-region is treated as an independent point cloud, and the aforementioned random RNN sampling method is utilized (as shown in Figure 6). Due to the varying sizes of each sub-region, the number of sampling seed points

differ. This paper proposes the following equation to determine the number of sampling seed points for each region.

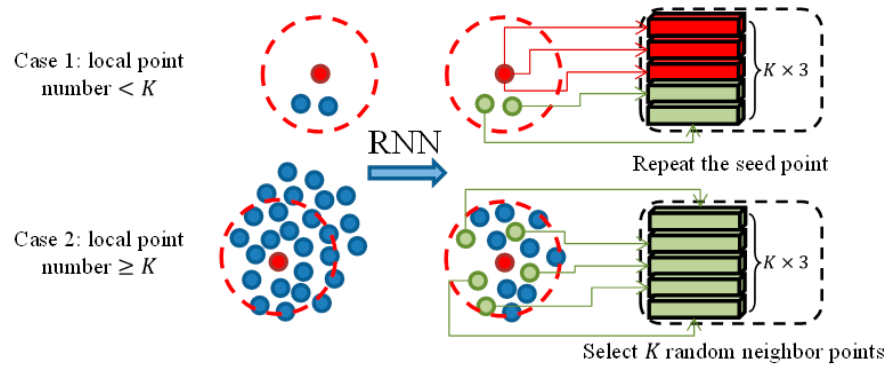


Figure 4. The RNN search scenarios of each seed point. Suppose $K = 5$; the seed point of case 1 contains just two neighbor points, so three duplicated instances of the seed point are added to obtain a 5×3 tensor. The seed point of case 2 contains 14 neighbor points, and five points are randomly picked from them to generate a 5×3 tensor.

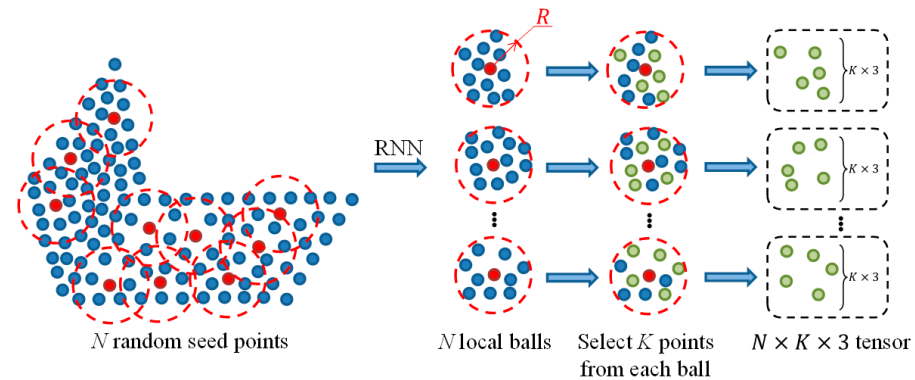


Figure 5. Workflow of the random RNN sampling method. In the last column, each green point represents a 1×3 tensor.

Suppose there are l regions, where l is less than or equal to N , and each region contains K or more points. After arranging these regions in descending order based on their point count, the number of points in the i -th region is denoted as N_i , and the undetermined seed point number in the i -th region is denoted as S_i . We have $N_1 \geq N_2 \geq \dots \geq N_l$. Then, S_i can be determined by Equation (1).

$$S_i = 1 + \lfloor \frac{N_i - 1}{\sum_{j=1}^l (N_j - 1)} \times (N - l) \rfloor + f(i) \tag{1}$$

where $f(k) = \begin{cases} 1, & \text{if } k \leq \left(N - l - \sum_{i=1}^l \lfloor \frac{N_i - 1}{\sum_{j=1}^l (N_j - 1)} \times (N - l) \rfloor \right) \\ 0, & \text{else} \end{cases}$. It is easy to verify that

(i) $\sum_{i=1}^l S_i = N$.

(ii) $N - l - \sum_{i=1}^l \lfloor \frac{N_i - 1}{\sum_{j=1}^l (N_j - 1)} \times (N - l) \rfloor = \sum_{i=1}^l \frac{N_i - 1}{\sum_{j=1}^l (N_j - 1)} \times (N - l) - \sum_{i=1}^l \lfloor \frac{N_i - 1}{\sum_{j=1}^l (N_j - 1)} \times (N - l) \rfloor$
 $\times (N - l) = \sum_{i=1}^l \left(\frac{N_i - 1}{\sum_{j=1}^l (N_j - 1)} \times (N - l) - \lfloor \frac{N_i - 1}{\sum_{j=1}^l (N_j - 1)} \times (N - l) \rfloor \right) \leq l$.

Thus, the proposed Equation (1) is reasonable.

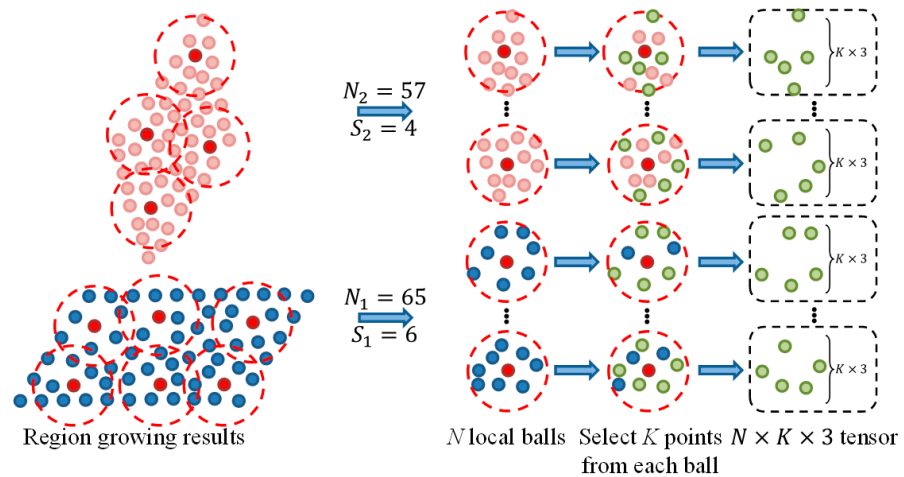
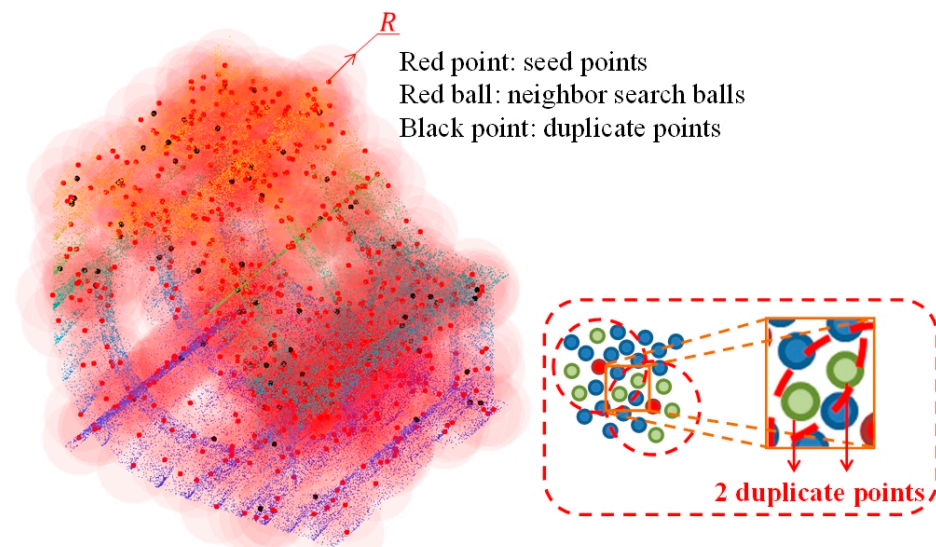


Figure 6. Workflow of the random RNN sampling method based on region growing results. The input point cloud is divided into two regions according to the region growing method. Suppose $K = 5$, $N = 10$, according to Equation (1), we have $S_1 = 6$ and $S_2 = 4$.

Through the use of these two methods, one can obtain the seed points (a $N \times 3$ tensor) and the neighbor points of these seed points (a $N \times K \times 3$ tensor). The $N \times K$ points are the actual perceived point cloud of the deep learning neural network. With the increase of N and K , the size of this point cloud can become quite large. These two methods will inevitably result in duplicate points in the perceived point cloud (as shown in Figure 7). Researchers can avoid this limitation by designing more sophisticated and complex methodologies. In this paper, on one hand, due to the relatively low proportion of the duplicate points, and on the other hand, for the purpose of ensuring fast execution, we accept the presence of these duplicate points. In addition, the random sampling method mentioned here can be replaced by the farthest point sampling (FPS) algorithm. But due to the slower execution speed of the FPS algorithm in dense point sets, we do not adopt it here.

Compared to the random RNN sampling method, the second sampling method considers the sub-region information of the seed points during the RNN search process; therefore, it may perform better in some tasks. Furthermore, the pre-segmentation part of the proposed module can be replaced by any traditional point cloud segmentation or grouping approach; for example, the super voxel growing method [28].



An instance: 163,840 points (containing 93 duplicate points)

Figure 7. An instance of duplicate points.

2.2. The Modified PointNet++ Module

The pre-segmentation results, N seed points (a $N \times 3$ tensor) and $N \times K$ neighbor points (a $N \times K \times 3$ tensor), are fed into the point-based deep learning neural network. We treat these input data as two separate components: one is a point cloud containing N points, and the other comprises N sub point clouds, each containing K points. The modified PointNet++ module aims to accomplish two tasks: (1) batch processing of the latter N sub point clouds, and (2) establishing a connection between these two data components.

Figure 8 illustrates the structure of the modified PointNet++ module. The module involves four parts.

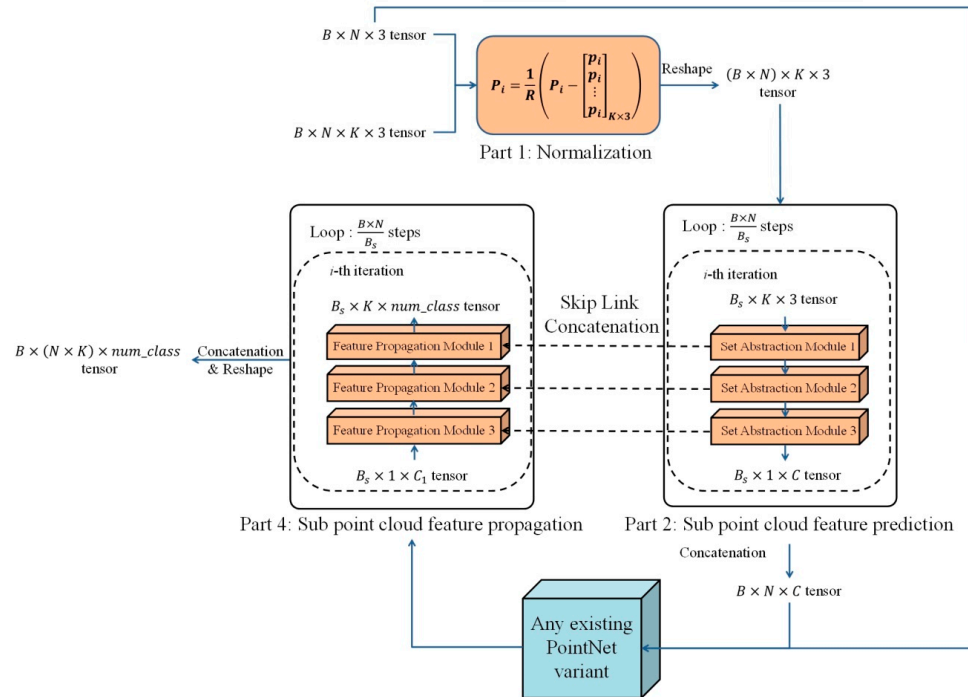


Figure 8. The modified PointNet++ module.

(1) All input sub point clouds are normalized into a unit sphere to ensure the feature presentation ability of the feature perceptron modules and the comparability of these sub point clouds. Supposing that the i -th sub point cloud is denoted by $\mathbf{P}_i \in \mathbf{R}^{K \times 3}$, and its corresponding seed point is denoted by $p_i \in \mathbf{R}^{1 \times 3}$, \mathbf{P}_i is normalized by Equation (2).

$$\mathbf{P}_i = \frac{1}{R} \left(\mathbf{P}_i - \begin{bmatrix} p_i \\ p_i \\ \vdots \\ p_i \end{bmatrix}_{K \times 3} \right) \quad (2)$$

where R is the search radius of the RNN sampling method mentioned in Section 2.1.

(2) Determine the batch size B_s of the sub point clouds. In general, for the sake of convenience, it is feasible to predefine the values of the variables B_s and N to fulfill $B_s | N$ (i.e., N is divisible by B_s). The calculation of local features for all sub point clouds is iteratively performed $\frac{N}{B_s}$ times, with the feature for each sub point cloud predicted using the standard, stacked PointNet++ set abstraction (SA) modules. In the case of multiple input raw point clouds (i.e., the input is a $B \times N \times 3$ tensor and a $B \times N \times K \times 3$ tensor, B is the batch size of raw point clouds), we can regard the number of sub point clouds as $B \times N$, and then calculate features through the aforementioned process.

(3) The features of the N sub point clouds are an $N \times C$ tensor, where C represents the number of the feature channel. If considering the multiple input raw point clouds, the

features will be a $(B \times N) \times C$ tensor, which can be reshaped as $B \times N \times C$ size. Either the $N \times C$ feature tensor or the $B \times N \times C$ feature tensor will serve as extra information, which will be jointly fed into any existing PointNet variant along with the $N \times 3$ point coordinate tensor or the $B \times N \times 3$ point coordinate tensor.

(4) (Optional) In some tasks, such as semantic segmentation, the feature propagation (FP) module corresponding to the SA modules of the sub point cloud is executed using an iterative process similar to '(2)'. In each iteration, the stacked full connection layers and drop out layers are carried out to reduce the GPU memory consumption.

By employing this modified PointNet++ module, the maximum number of points processed in real time within the whole deep learning network is reduced to $\max(B \times N, B_s \times K)$ (this value is $B \times N \times K$ in the standard PointNet variants), while the farthest point sampling (FPS) algorithm is now capable of handling a maximum of $\max(N, K)$ points (this value is $N \times K$ in the standard PointNet variants). This leads to a reduction in GPU memory consumption of the SA modules and the FP modules during the processing of dense point sets. In the case of recursively utilizing this module (as shown in Figure 9), people can calculate a larger point set with a smaller GPU memory.

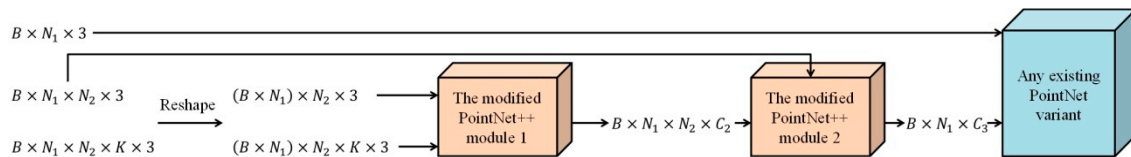


Figure 9. Simplified illustration of recursively utilizing the modified PointNet++ module. In this network, the maximum number of points processed in real-time is reduced to $\max(B \times N_1, B_s \times K, B_s \times N_2)$, and the farthest point sampling algorithm is now capable of handling a maximum of $\max(N_1, N_2, K)$ points.

The integration of the modified PointNet++ module with any existing PointNet variant can be easily accomplished. For instance, in Section 3, it is connected with PointNet++, DGCNN and the Partial Registration Network (PRnet) [29]. Essentially, the modified PointNet++ module serves as a complementary deep learning network structure to the pre-segmentation methods proposed in Section 2.1. Consequently, when the modified PointNet++ module is combined with any existing method, the loss function remains consistent with the aggregated existing method.

3. Experimental Results

The proposed module is integrated into PointNet++ and DGCNN to evaluate its semantic segmentation performance. In addition, the applications of the proposed module in other deep learning tasks are tested to show the feasibility of the proposed module in various PointNet variants.

3.1. Dataset

All deep learning networks in this paper are evaluated on four datasets:

(i) ModelNet10 [20]: ModelNet10 dataset is a collection of 3D meshed models used for the object classification task in computer vision and machine learning research. It contains 4899 CAD models from 10 categories (toilets, chairs, nightstands, bookshelves, dressers, beds, tables, sofas, desks and monitors). We utilize a subset containing four random categories of ModelNet10 to test the proposed module. In the subset, 1000 samples are used for training and 290 samples for testing.

(ii) Stanford 3D semantic parsing dataset [30]: Stanford 3D semantic parsing dataset includes point clouds and corresponding semantic labels for six areas of 271 indoor scenes. The semantic labels of all these scenes are divided into 13 categories, such as chairs, walls, ceilings, tables, and some other structural elements.

(iii) Our hull block dataset: 149 standard CAD models of ship hull blocks are gathered. They are divided into four categories (bottom blocks, side shell blocks, deck blocks, fore and aft blocks, note that 'fore and aft block' is one category) and split into 120 training samples and 29 testing samples.

(iv) Our hull block butt-joint part dataset: In the ship construction step, all hull blocks are joined together with butt joints; thus, the construction quality control of hull block butt-joints plays a significant role in shipyards. We gather 74 hull block butt-joint part point clouds from CAD models and realistic scanning scenes. Each point cloud contains semantic labels from four categories (hull plates, T-steels, flat steels and bulb flat steels). All point clouds are split into 60 training samples and 14 testing samples.

In the pre-processing step, the neighbor search radius is set to 0.2 after the point clouds are rescaled into a unit sphere. In the region growing configuration, the minimum point number and maximum point number of each region are set to 800 and 100,000, the neighbor point number is set to 30, the threshold of normal vectors to determine whether to add the neighbor point to the region of the central point is set to 15° , the curvature threshold which determines whether to consider the point in the region growing step is set to 0.05. According to these configurations, each input point cloud can be divided into sub-regions.

The existing PointNet variants adopt their default configurations as presented in the original papers. When the proposed module is taken into consideration, we designate the variables B , N , K and B_s as 5, 640, 256, and 50, respectively, which means that for each model or point cloud in the four aforementioned datasets, 163,840 points are sampled to represent the input 3D geometries. Therefore, in order to ensure sampling quality, we require the point number in each instance of the datasets to be greater than 163,840. This point number threshold is set to 300,000. For the Stanford 3D semantic parsing dataset, unfortunately, some scenes have to be discarded as their point numbers are less than 300,000, and the final number of instances remaining is 199. These 199 scenes are split into 160 training samples and 39 testing samples. Readers may easily notice that the processing point number of the proposed module (163,840 points) is significantly larger than the point numbers in the original PointNet variants (1024 or 2048 points). Each sampled point cloud is rescaled into a unit sphere, and just their 3D coordinates are utilized as input data; the remaining information, such as normal vectors, colors, etc., is disregarded. In the training step, all data are augmented by randomly rotating, translating, scaling and perturbing the point coordinates.

3.2. Semantic Segmentation

To test the semantic segmentation performance of the proposed module, it is embedded into PointNet++ and DGCNN, and evaluated on the Stanford 3D semantic parsing dataset and our hull block butt-joint part dataset. As the Kernel Point Convolution (KPconv) [31] can process more points than the original PointNet++ and DGCNN, it is also evaluated in this section. The architecture of the proposed module is set as the configuration shown in Figures 10 and 11 to meet the input $5 \times 640 \times 256 \times 3$ tensor procedure.

The set abstraction part of the proposed module is shown in Figure 10. Since the SA modules are utilized to generate the feature of the $50 \times 256 \times 3$ tensor, the sample point number of the FPS method in the first SA module is set as 64, the local search number and radius are set to 32 and 0.3, and the layer sizes of the multi-layer perceptron are set to (64, 128), which means that the multi-layer perceptron consists of two fully connected layers — the first layer contains 64 units and the second layer contains 128 units. The second SA module is similar to the first SA module. The FPS sample point number is set to 32, the local search number is set to 16, the local search radius is set to 0.5, and the layer sizes of the multi-layer perceptron are set to (128, 128). As for the third SA module, all local points and features are aggregated to compute the feature of the seed points, so the FPS and the local neighbor search procedures are abandoned, and the layer sizes of multi-layer perceptron are set to (128, 128).

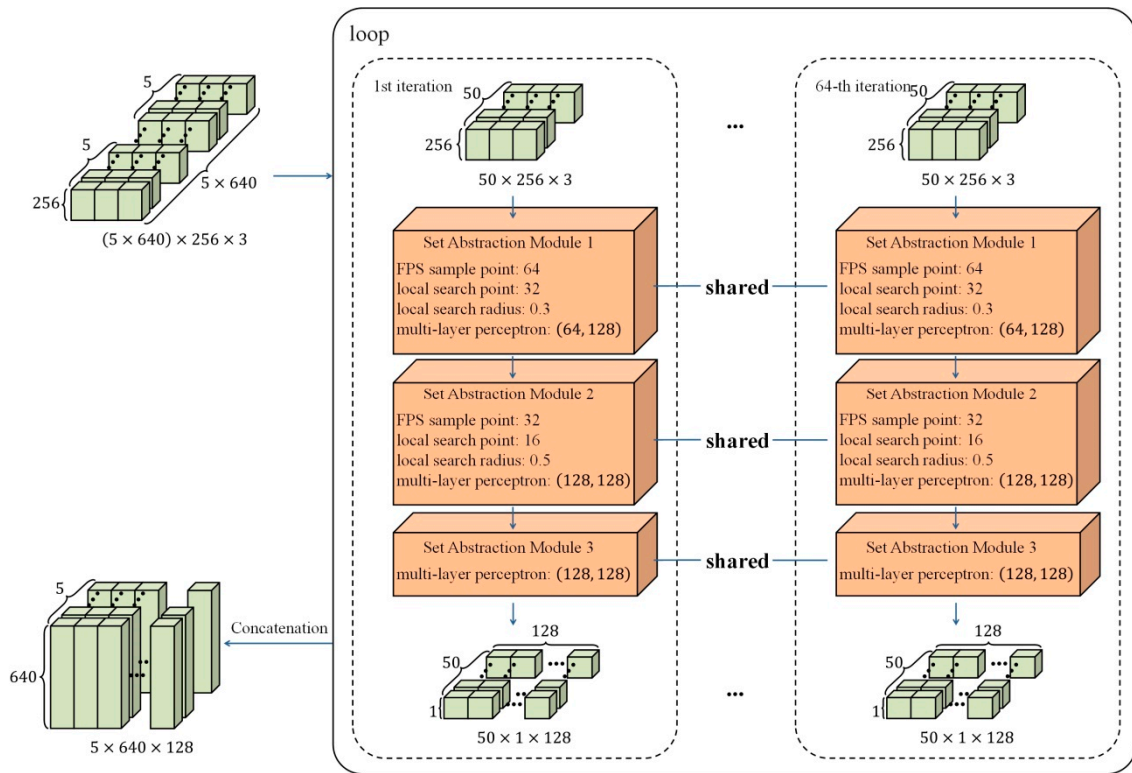


Figure 10. Configuration of the set abstraction part of the proposed module.

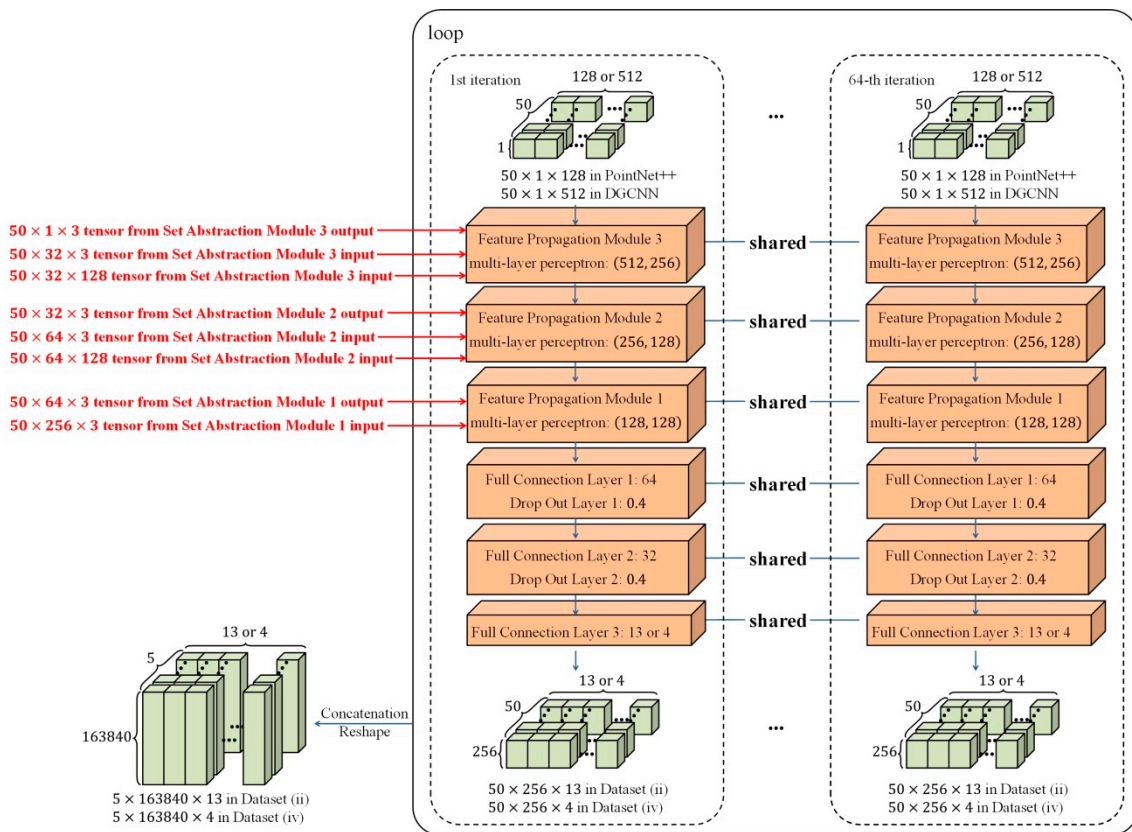


Figure 11. Configuration of the feature propagation part of the proposed module.

The PointNet++ or the DGCNN is subsequently connected to the proposed module. Their input data are a $5 \times 640 \times 3$ tensor and a $5 \times 640 \times 128$ tensor, which can be treated as the 3D coordinates and corresponding features of five mini point clouds, and each point cloud contains 640 points. As the input point number is different from the original PointNet++ or DGCNN, some parameters of these two PointNet variants require modification. The parameters of three SA modules in the PointNet++ are set as shown in Figure 12, and their explanation is similar to that of the proposed module, hence there is no need to reiterate it. As for DGCNN, due to the limited association between its network architecture and the input point number, we only make slight modifications to its parameters; the local search number is set to 64, and the final embedded channel is set to 512.

The feature propagation part of the proposed module is shown in Figure 11. The feature propagation architecture contains three standard feature propagation modules, three full connection layers and two drop out layers. During each iteration, if the preceding network is PointNet++, the feature propagation architecture receives a $50 \times 1 \times 128$ tensor. However, if the preceding network is DGCNN, the feature propagation architecture receives a $50 \times 1 \times 512$ tensor, as its final embedded channel is set to 512. The first FP module is denoted as ‘Feature Propagation Module 3’ since it gathers the input points, input features, and output points of ‘Set Abstraction Module 3’, illustrated in Figure 10 through a skip link concatenation procedure. The second FP module utilizes the input points, input features, and output points of ‘Set Abstraction Module 2’, as well as the output features of the first FP module, to generate high-level features. As just the coordinates of each dataset are used, the final FP module only gathers the input points and output points of ‘Set Abstraction Module 1’. The output features are processed by ‘Full Connection Layer 1’, which consists of 64 units, ‘Drop Out Layer 1’, which may set the output value of each neuron in the preceding layer to zero with probability 0.4, ‘Full Connection Layer 2’, ‘Drop Out Layer 2’, and the output layer ‘Full Connection Layer 3’. If the whole network is evaluated on the Stanford 3D semantic parsing dataset, the output channel is 13. If the whole network is tested on our hull block butt-joint part dataset, the output channel is 4.

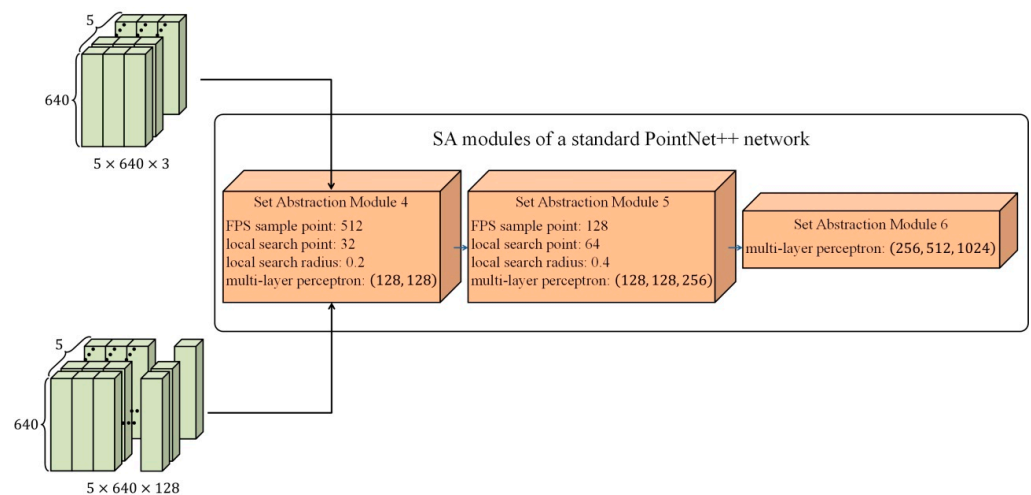


Figure 12. Configuration of the set abstraction part of the standard PointNet++ network after the set abstraction part of the proposed module.

Similar to the configuration in Figure 12, the three feature propagation modules of the original PointNet++ are modified to an FP module with a multi-layer perceptron (512, 256) (with two full connection layers; the first layer contains 512 neurons, while the second layer includes 256 neurons), an FP module with a multi-layer perceptron (256, 128) and an FP module with a multi-layer perceptron (256, 128).

As for KPconv, each point cloud is down-sampled into 81,920 points. The super parameters of KPconv are consistent with that in its original work. Note that the original work of KPconv achieves the segmentation of dense point clouds by dividing the whole

point cloud into unrelated sub-point clouds in many random sphere regions. We may certainly adhere to this approach to deal with a dense or even infinite number of points, but the maximum point number that KPConv can actually handle should be the max point number of the sub point clouds.

In the training step, the Adam [32] algorithm is applied to optimize the networks containing the proposed modified PointNet++ module. The learning rate is set to 0.001, the weight decay rate is set to 0.0001, the momentum value of batch-normalization is 0.9, β_1 and β_2 are set to their default values 0.9 and 0.999, and the data batch size is 5.

Tables 1 and 2 show the semantic segmentation results of seven methods (original PointNet++, PointNet++ with our random RNN sampling, PointNet++ with our random RNN sampling based on region growing results, original DGCNN, DGCNN with our random RNN sampling, DGCNN with our random RNN sampling based on region growing results, and KPConv) on the hull block butt-joint part dataset and the Stanford 3D semantic parsing dataset. Compared to the original PointNet++ and DGCNN, the incorporation of our proposed modules in PointNet++ and DGCNN yields superior performance. For PointNet++ on the Stanford 3D semantic parsing dataset, the improvements from both random RNN sampling pre-processing and region growing pre-processing are similar (as shown in Table 2, 86.7% and 86.2%), while for other cases, the improvement effect of the region growing pre-processing module surpasses the random RNN sampling pre-processing module. The advantage of our region growing pre-processing module may potentially be attributed to its incorporation of implicit prior region segmentation knowledge, where the ground truth semantic segmentation information is generally closely associated with the region segmentation information. Figures 13 and 14 show four semantic segmentation instances from our hull block butt-joint part dataset and the Stanford 3D semantic parsing dataset, respectively. In the two figures, each column represents one instance, different semantic labels are represented by different colors, and the overall prediction accuracy of each instance is shown in the title of its corresponding subplot. Our pre-processing module outperforms KPConv in our hull block butt-joint part dataset, and achieves similar segmentation accuracy of KPConv in the Stanford 3D semantic parsing dataset. Compared with the sparse representation of 2048 points in the original PointNet++ and DGCNN, the 163,840 points processed by the proposed modules preserve a more comprehensive range of the original point cloud information, and its geometric structure closely resembles that of the original point cloud, which may be an attractive advantage of our work.

To analyze the properties of our region growing preprocessing module, the point intersection over union (IoU) results on our hull block butt-joint part dataset are considered. As shown in Table 3, the methods embedding our random RNN sampling based on region growing results outperform the methods embedding our random RNN sampling in the segmentation of the 'hull plate' and 'T-steel' classes. As most points in the hull block butt-joint part dataset belong to these two classes, the overall accuracies of the corresponding methods in Table 1 are better. As for the Stanford 3D semantic parsing dataset, as shown in Table 4, the methods embedding our random RNN sampling based on region growing results outperform the methods embedding our random RNN sampling in most classes. However, in some classes, like 'beam', 'ceiling', 'sofa' and 'window', the latter outperform the former; since these classes contain a considerable number of points in the whole point clouds, the overall accuracies of the former in Table 2 are sometimes superior to the latter, and sometimes inferior to the latter.

Table 1. Results of 3D semantic segmentation in our hull block butt-joint part dataset.

Deep Learning Method	Train Overall Accuracy	Test Overall Accuracy
PointNet++	88.5%	86.9%
PointNet++ with random RNN sampling	87.1%	80.4%
PointNet++ with region growing results	94.0%	89.7%
DGCNN	81.9%	79.0%
DGCNN with random RNN sampling	86.7%	79.8%
DGCNN with region growing results	93.8%	89.1%
KPConv	89.6%	87.4%

Table 2. Results of 3D semantic segmentation in the Stanford 3D semantic parsing dataset.

Deep Learning Method	Train Overall Accuracy	Test Overall Accuracy
PointNet++	83.8%	84.2%
PointNet++ with random RNN sampling	90.6%	86.7%
PointNet++ with region growing results	90.2%	86.2%
DGCNN	90.8%	84.0%
DGCNN with random RNN sampling	90.2%	83.4%
DGCNN with region growing results	92.5%	86.2%
KPConv	93.5%	86.5%

Table 3. IoU results in our hull block butt-joint part dataset.

Deep Learning Method	Hull Plate	T-Steel	Flat Steel	Bulb Flat Steel
PointNet++ with random RNN sampling	64.4	69.0	65.6	73.0
PointNet++ with region growing results	84.3	71.5	23.9	60.2
DGCNN with random RNN sampling	66.6	59.7	47.1	50.4
DGCNN with region growing results	92.7	83.7	52.5	28.0

Table 4. IoU results in the Stanford 3D semantic parsing dataset.

Deep Learning Method	Beam	Board	Book.	Ceil.	Chair	Clut.	Col.	Door	Floor	Sofa	Table	Wall	Wind.
PointNet++ with random RNN sampling	77.2	48.8	43.2	89.0	63.1	46.5	42.0	62.3	92.4	70.0	61.7	73.0	42.1
PointNet++ with region growing results	73.5	51.2	62.9	90.0	71.1	49.7	75.4	64.9	94.2	60.6	66.0	77.0	65.8
DGCNN with random RNN sampling	87.8	58.9	64.2	91.5	62.0	49.0	63.5	59.7	94.2	80.1	67.7	75.8	81.1
DGCNN with region growing results	85.0	73.4	65.8	90.8	70.6	50.6	65.9	63.7	94.7	86.5	73.7	76.8	75.9

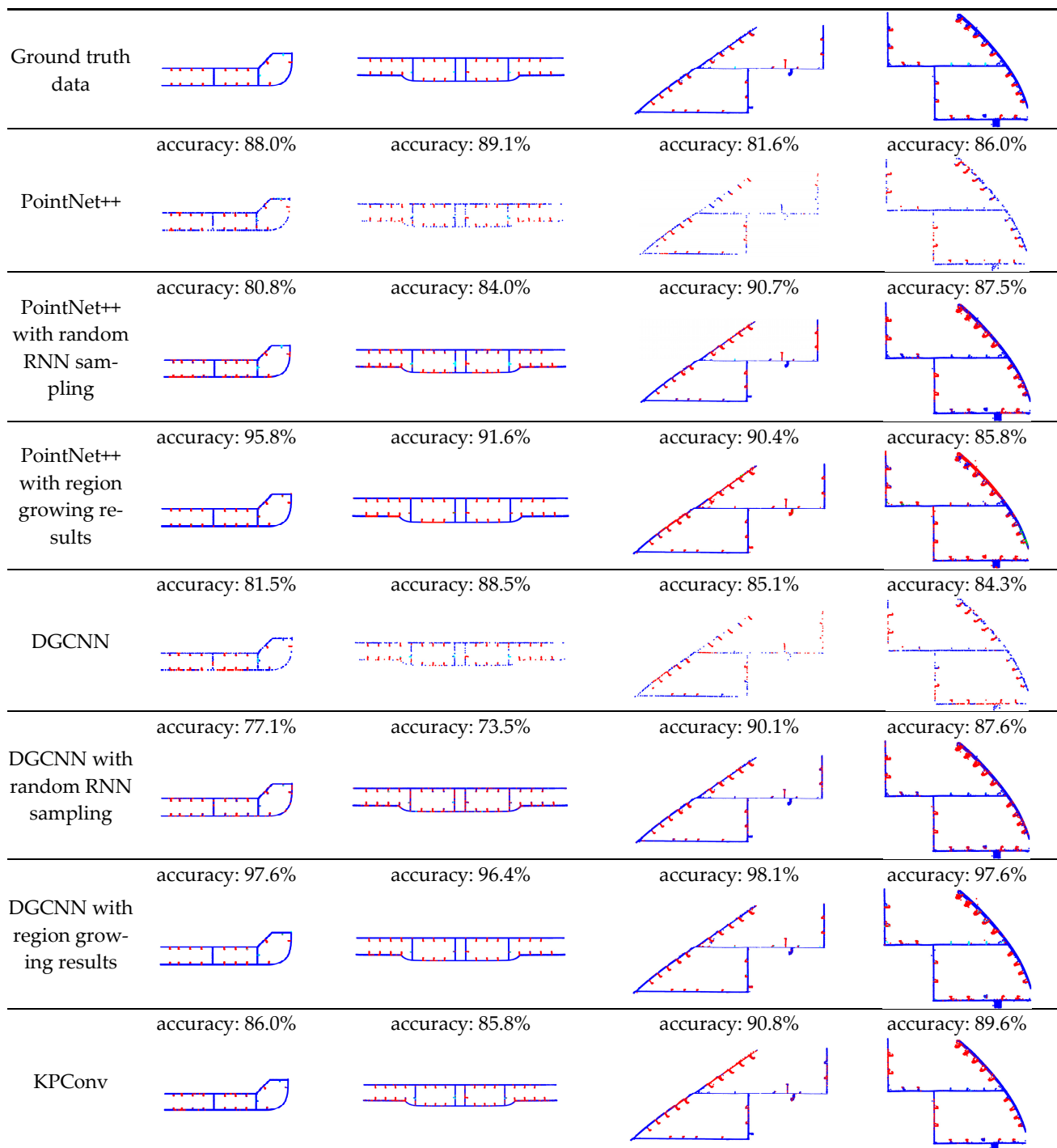


Figure 13. Four semantic segmentation instances in our hull block butt-joint part dataset.

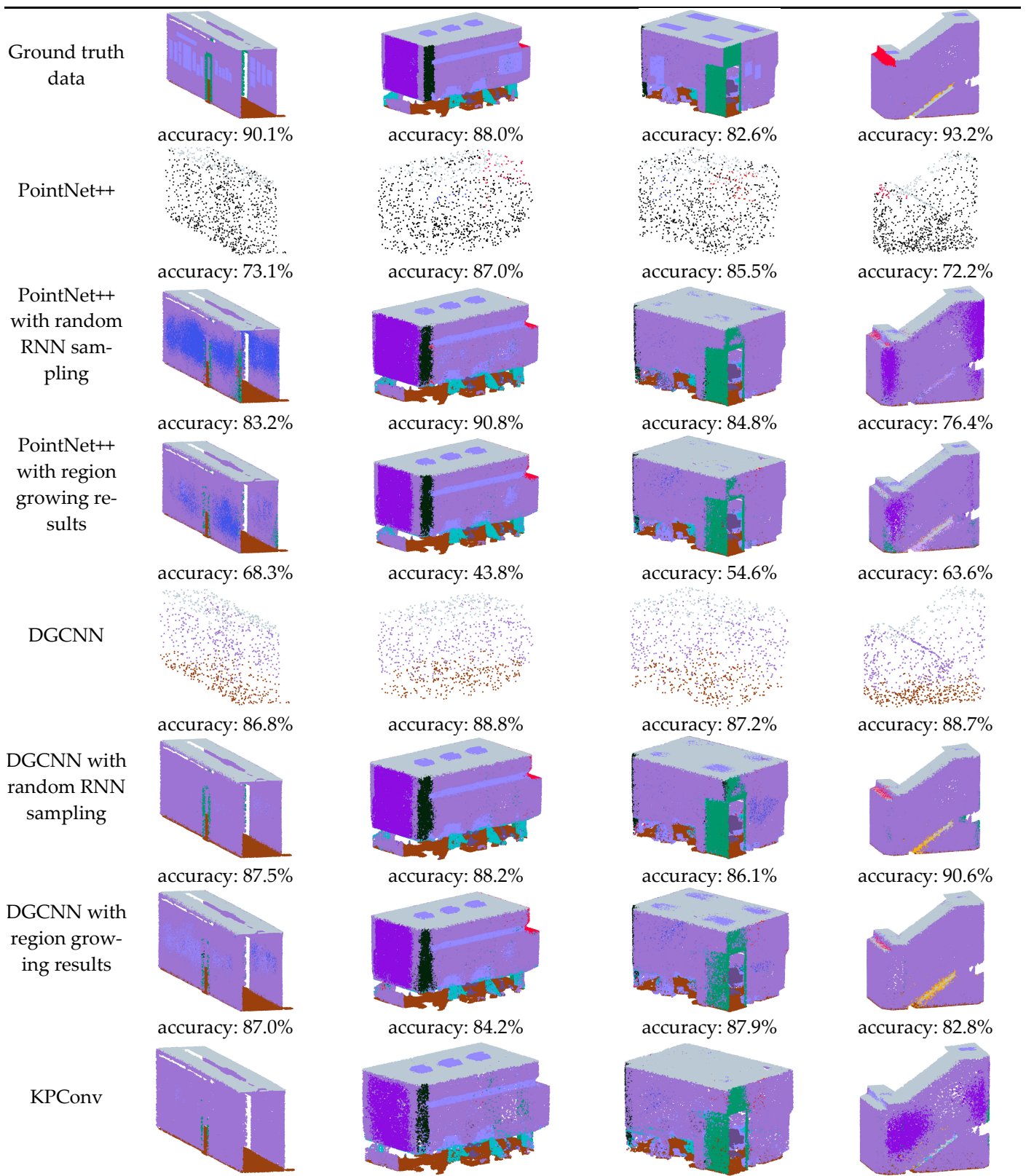


Figure 14. Four semantic segmentation instances in the Stanford 3D semantic parsing dataset.

3.3. Other Applications

To show that the proposed module is applicable in different point-based deep learning tasks, this section utilizes the proposed module to process two other deep learning applications, classification and 3D registration, both with dense input points.

In the classification task, the proposed module is also embedded into PointNet++ and DGCNN, and evaluated in the ModelNet10 dataset and our hull block dataset. The architecture of the classification networks is similar, as shown in Figures 10 and 12. The training configuration is the same as that of the semantic segmentation experiment part in Section 3.2.

Tables 5 and 6 show the classification results of six methods in our hull block dataset and ModelNet10 dataset. It can be found that four test accuracies in Table 5 are the same, which is caused by the scarcity of our hull block dataset. The proposed module outperforms the original PointNet++ and DGCNN in Table 5. The classification ability of the proposed module is similar to the original PointNet++ and DGCNN (as shown in Table 6, PointNet++ with random RNN sampling achieves 95.1% test accuracy, which is close to 95.2% in PointNet++, and DGCNN with region growing results achieves 98.3% test accuracy, which outperforms 96.0% in DGCNN) in ModelNet10. Tables 5 and 6 illustrate the feasibility of the proposed module in the classification task of dense input points (each input point cloud contains 163,840 points).

Table 5. Results of 3D object classification in our hull block dataset.

Deep Learning Method	Train Accuracy	Test Accuracy
PointNet++	97.5%	93.1%
PointNet++ with random RNN sampling	98.3%	96.6%
PointNet++ with region growing results	97.5%	96.6%
DGCNN	90.8%	86.2%
DGCNN with random RNN sampling	98.3%	96.6%
DGCNN with region growing results	98.3%	96.6%

Table 6. Results of 3D object classification in ModelNet10.

Deep Learning Method	Train Accuracy	Test Accuracy
PointNet++	97.4%	95.2%
PointNet++ with random RNN sampling	96.5%	95.1%
PointNet++ with region growing results	94.7%	94.8%
DGCNN	97.8%	96.0%
DGCNN with random RNN sampling	98.3%	96.9%
DGCNN with region growing results	98.4%	98.3%

As for the 3D registration task, the proposed module is embedded into PRnet, and evaluated in our hull block dataset and the ModelNet10 dataset.

PRnet is a deep learning network designed for a partial-to-partial point cloud registration task. PRnet is self-supervised, jointly learning an appropriate geometric representation, a key point detector that finds points in common between partial views, and key point-to-key point correspondences [28]. The architecture illustrated in Figure 10 is used to replace the original prediction point embedding head of PRnet, and the remaining parts of PRnet follows their original configurations.

Tables 7 and 8 show the 3D registration results of three methods (PRnet, PRnet with our random RNN sampling, and PRnet with our random RNN sampling based on region growing results) in our hull block dataset and ModelNet10. Figures 15 and 16 show four registration instances from the two datasets. In the two figures, each row represents one instance, the initial positions and final poses after alignment of the fixed-point clouds and the float point clouds are shown; in each subplot, the purple point cloud represent the fixed-point cloud, while the green point cloud represent the float point cloud. Similar to

the classification results, the utilization of the proposed module empowers PRnet with the capability to process dense input point clouds.

Table 7. Results of 3D registration in our hull block dataset.

Deep Learning Method	Train Total Loss	Test Total Loss
PRnet	0.28	0.24
PRnet with random RNN sampling	0.15	0.16
PRnet with region growing results	0.18	0.16

Table 8. Results of 3D registration in the ModelNet10 dataset.

Deep Learning Method	Train Total Loss	Test Total Loss
PRnet	0.27	0.26
PRnet with random RNN sampling	0.17	0.17
PRnet with region growing results	0.16	0.15

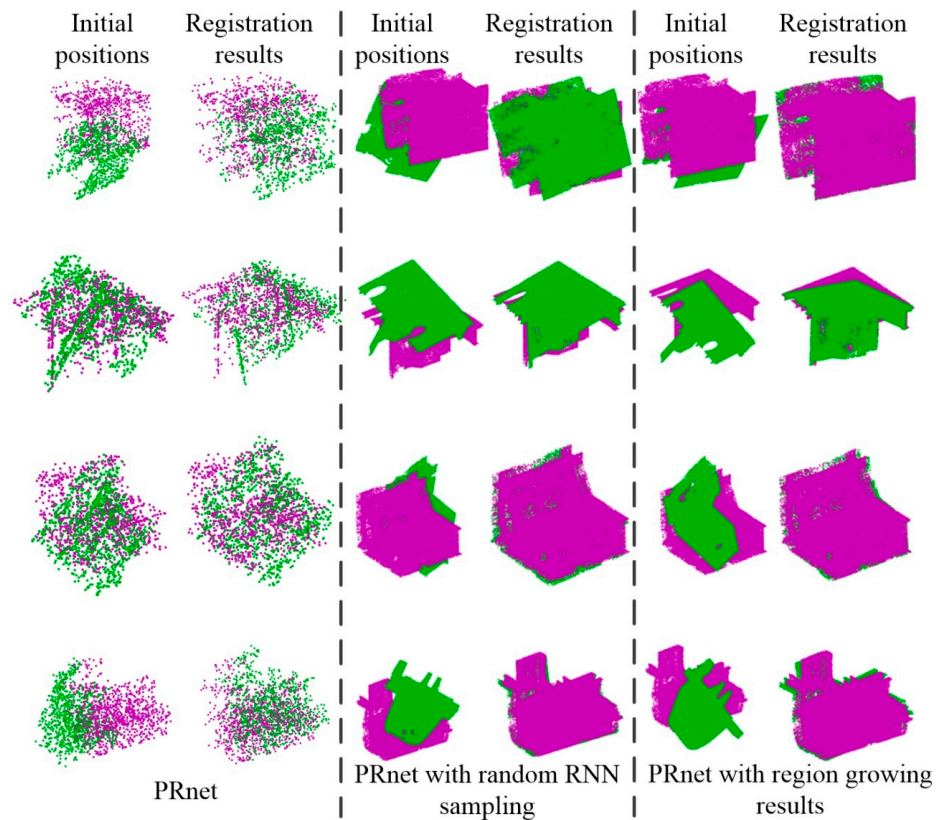


Figure 15. Four registration instances in our hull block dataset.

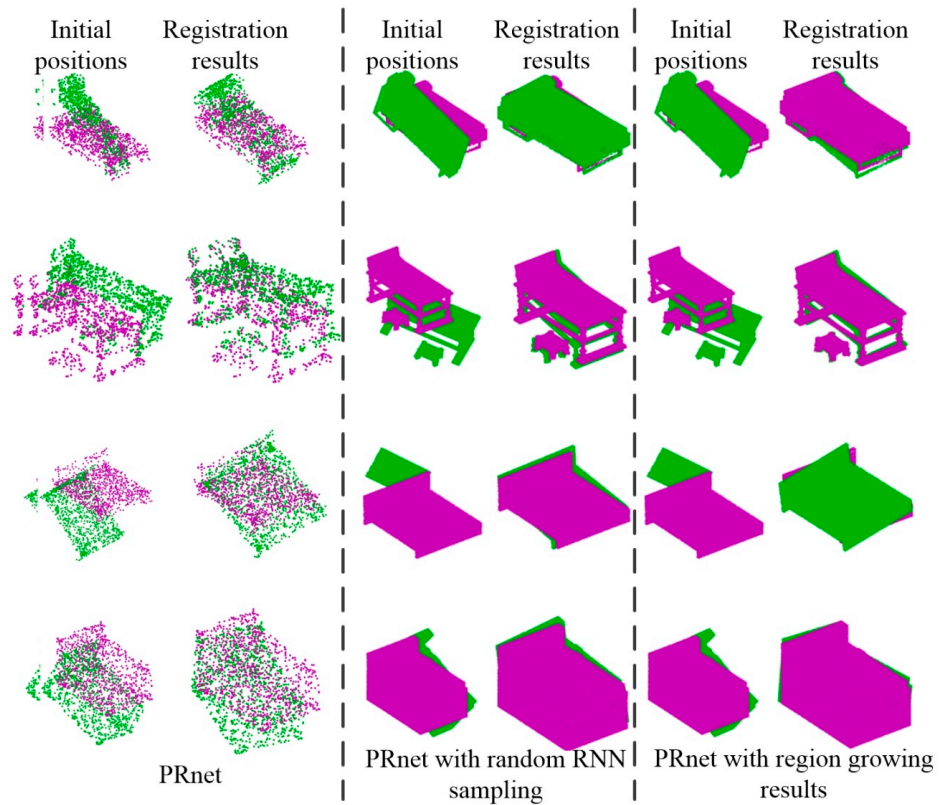


Figure 16. Four registration instances in the ModelNet10 dataset.

4. Conclusions

To process general dense point clouds in the ship engineering field by point-based deep learning methods, this paper presents a pre-processing module which utilizes the traditional point cloud processing methods and the PointNet++ paradigm. We evaluate the semantic segmentation performance, classification performance, and 3D registration performance of the proposed module on two ship structure datasets, our hull block dataset and our hull block butt-joint part dataset, and two popular point cloud datasets, the ModelNet10 dataset and the Stanford 3D semantic parsing dataset. The experimental results show that:

(i) In the point-based semantic segmentation task, the proposed modules achieve 89.7% and 89.1% overall accuracies on our hull block butt-joint part dataset, which outperform the original PointNet++, DGCNN, and KPConv. In the Stanford 3D semantic parsing dataset, the proposed modules achieve 86.7% and 86.2% overall accuracies, which outperform the original PointNet++ and DGCNN, and achieve a similar performance of 86.5% accuracy in KPConv.

(ii) In the 3D classification task, the proposed module obtains 96.6% accuracy, which outperforms the original PointNet++ and DGCNN in our hull block dataset. In ModelNet10, PointNet++ with random RNN sampling achieves a 95.1% test accuracy, which is close to 95.2% in PointNet++, and DGCNN with region growing results achieves a 98.3% test accuracy, which outperforms 96.0% in DGCNN.

(iii) In the 3D registration task, the utilization of the proposed module empowers PRnet with the capability to process dense input point clouds.

The proposed module is highly applicable to the semantic segmentation of dense input clouds, and may provide technical support to some further complex procession of dense point clouds based on the semantic segmentation results. The proposed module may provide a useful semantic segmentation tool for realistic dense point clouds in various industrial applications.

Author Contributions: Conceptualization, S.H., Y.L. and J.W.; Methodology, S.H.; Software, S.H.; Validation, S.H.; Investigation, S.H.; Data curation, S.H.; Writing—original draft, S.H.; Writing—review & editing, S.H.; Visualization, J.W.; Supervision, Y.L., J.W., R.L., X.L. and J.S.; Funding acquisition, J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (51979033) and the Dalian Science and Technology Innovation Foundation (2019J12GX021).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

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

References

1. Wang, J.; Huo, S.; Liu, Y.; Li, R.; Liu, Z. Research of Fast Point Cloud Registration Method in Construction Error Analysis of Hull Blocks. *Int. J. Nav. Archit. Ocean Eng.* **2020**, *12*, 605–616. [[CrossRef](#)]
2. Wei, Y.; Ding, Z.; Huang, H.; Yan, C.; Huang, J.; Leng, J. A Non-Contact Measurement Method of Ship Block Using Image-Based 3D Reconstruction Technology. *Ocean Eng.* **2019**, *178*, 463–475. [[CrossRef](#)]
3. Zhang, W.; Wu, Y.; Tian, X.; Bao, W.; Yu, T.; Yang, J. Application Research of Ship Overload Identification Algorithm Based on Lidar Point Cloud. In Proceedings of the 2022 2nd International Conference on Electrical Engineering and Mechatronics Technology (ICEEMT), Hangzhou, China, 1–3 July 2022; pp. 377–381.
4. Lu, X.; Li, Y.; Xie, M. Preliminary Study for Motion Pose of Inshore Ships Based on Point Cloud: Estimation of Ship Berthing Angle. *Measurement* **2023**, *214*, 112836. [[CrossRef](#)]
5. Wen, Y.; Chen, X.; Liu, C.; Liu, K. Research on Ship 3D Target Detection Method Based on Lidar Point Clouds. In Proceedings of the International Conference on Mechanisms and Robotics (ICMAR 2022), Zhuhai, China, 25–27 February 2022; pp. 103–108.
6. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 652–660.
7. Maturana, D.; Scherer, S. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–3 October 2015; pp. 922–928.
8. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; Volume 30.
9. Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.E.; Bronstein, M.M.; Solomon, J.M. Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. Graph.* **2019**, *38*, 1–12. [[CrossRef](#)]
10. Yang, Z.; Sun, Y.; Liu, S.; Jia, J. 3DSSD: Point-Based 3D Single Stage Object Detector. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 11037–11045.
11. Zhang, Y.; Hu, Q.; Xu, G.; Ma, Y.; Wan, J.; Guo, Y. Not All Points Are Equal: Learning Highly Efficient Point-Based Detectors for 3D LiDAR Point Clouds. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 18931–18940.
12. Shi, S.; Jiang, L.; Deng, J.; Wang, Z.; Guo, C.; Shi, J.; Wang, X.; Li, H. PV-RCNN++: Point-Voxel Feature Set Abstraction with Local Vector Representation for 3D Object Detection. *Int. J. Comput. Vis.* **2023**, *131*, 531–551. [[CrossRef](#)]
13. Wang, Y.; Solomon, J. Deep Closest Point: Learning Representations for Point Cloud Registration. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 3522–3531.
14. Wu, Y.; Zhang, Y.; Fan, X.; Gong, M.; Miao, Q.; Ma, W. INENet: Inliers Estimation Network with Similarity Learning for Partial Overlapping Registration. *IEEE Trans. Circuits Syst. Video Technol.* **2023**, *33*, 1413–1426. [[CrossRef](#)]
15. Wang, H.; Liu, Y.; Dong, Z.; Wang, W. You Only Hypothesize Once: Point Cloud Registration with Rotation-Equivariant Descriptors. In Proceedings of the 30th ACM International Conference on Multimedia, ACM, Lisboa, Portugal, 10–14 October 2022; pp. 1630–1641.
16. Zhang, K.; Hao, M.; Wang, J.; de Silva, C.W.; Fu, C. Linked Dynamic Graph CNN: Learning on Point Cloud via Linking Hierarchical Features. *arXiv* **2019**, arXiv:1904.10014.
17. Huang, C.-Q.; Jiang, F.; Huang, Q.-H.; Wang, X.-Z.; Han, Z.-M.; Huang, W.-Y. Dual-Graph Attention Convolution Network for 3-D Point Cloud Classification. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–13. [[CrossRef](#)] [[PubMed](#)]
18. Zhang, C.; Wan, H.; Shen, X.; Wu, Z. PatchFormer: An Efficient Point Transformer with Patch Attention. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 19–20 June 2022; pp. 11789–11798.

19. Zhao, H.; Jiang, L.; Jia, J.; Torr, P.H.S.; Koltun, V. Point Transformer. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 11799–11808.
20. Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; Xiao, J. 3D ShapeNets: A Deep Representation for Volumetric Shapes. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1912–1920.
21. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision Meets Robotics: The Kitti Dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [[CrossRef](#)]
22. Caesar, H.; Bankiti, V.; Lang, A.H.; Vora, S.; Liong, V.E.; Xu, Q.; Krishnan, A.; Pan, Y.; Baldan, G.; Beijbom, O. nuScenes: A Multimodal Dataset for Autonomous Driving. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 11618–11628.
23. Zeng, A.; Song, S.; Nießner, M.; Fisher, M.; Xiao, J.; Funkhouser, T. 3Dmatch: Learning Local Geometric Descriptors from Rgb-D Reconstructions. *arXiv* **2017**, arXiv:1603.08182.
24. Friedman, J.H.; Bentley, J.L.; Finkel, R.A. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. Math. Softw.* **1977**, *3*, 209–226. [[CrossRef](#)]
25. Ram, P.; Sinha, K. Revisiting Kd-Tree for Nearest Neighbor Search. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 1378–1388.
26. Durovsky, F. Point Cloud Based Bin Picking: Object Recognition and Pose Estimation Using Region Growing Segmentation Algorithm. *Appl. Mech. Mater.* **2015**, *791*, 189–194. [[CrossRef](#)]
27. Hackel, T.; Wegner, J.D.; Schindler, K. Fast Semantic Segmentation of 3D Point Clouds with Strongly Varying Density. In Proceedings of the 2016 ISPRS Annual Congress of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Prague, Czech Republic, 12–19 July 2016; Volume III–3, pp. 177–184.
28. Lin, Y.; Wang, C.; Zhai, D.; Li, W.; Li, J. Toward Better Boundary Preserved Supervoxel Segmentation for 3D Point Clouds. *ISPRS J. Photogramm. Remote Sens.* **2018**, *143*, 39–47. [[CrossRef](#)]
29. Wang, Y.; Solomon, J.M. PRNet: Self-Supervised Learning for Partial-to-Partial Registration. In Proceedings of the Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019; Volume 32, pp. 8812–8824.
30. Armeni, I.; Sener, O.; Zamir, A.R.; Jiang, H.; Brilakis, I.; Fischer, M.; Savarese, S. 3D Semantic Parsing of Large-Scale Indoor Spaces. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 1534–1543.
31. Thomas, H.; Qi, C.R.; Deschaud, J.-E.; Marcotegui, B.; Goulette, F.; Guibas, L.J. KPConv: Flexible and Deformable Convolution for Point Clouds. *arXiv* **2019**, arXiv:1904.08889.
32. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.