Point Cloud Processing Methods for 3D Point Cloud Detection Tasks

WANG Chongchong[1], LI Yao[2], WANG Beibei[3], CAO Hong[3], ZHANG Yanyong[2]

(1. Anhui University, Hefei 230601, China;
2. University of Science and Technology of China, Hefei 230026, China;
3. Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, Hefei 230026, China)

DOI: 10.12142/ZTECOM.202304005

http://kns.cnki.net/kcms/detail/34.1294.TN.20231123.0900.002.html, published online November 23, 2023

Manuscript received: 2023-10-07

Abstract: Light detection and ranging (LiDAR) sensors play a vital role in acquiring 3D point cloud data and extracting valuable information about objects for tasks such as autonomous driving, robotics, and virtual reality (VR). However, the sparse and disordered nature of the 3D point cloud poses significant challenges to feature extraction. Overcoming limitations is critical for 3D point cloud processing. 3D point cloud object detection is a very challenging and crucial task, in which point cloud processing and feature extraction methods play a crucial role and have a significant impact on subsequent object detection performance. In this overview of outstanding work in object detection from the 3D point cloud, we specifically focus on summarizing methods employed in 3D point cloud processing. We introduce the way point clouds are processed in classical 3D object detection algorithms, and their improvements to solve the problems existing in point cloud processing. Different voxelization methods and point cloud sampling strategies will influence the extracted features, thereby impacting the final detection performance.

Keywords: point cloud processing; 3D object detection; point cloud voxelization; bird's eye view; deep learning

Citation (Format 1): WANG C C, LI Y, WANG B B, et al. Point cloud processing methods for 3D point cloud detection tasks [J]. ZTE Communications, 2023, 21(4): 38 – 46. DOI: 10.12142/ZTECOM.202304005

Citation (Format 2): C. C. Wang, Y. Li, B. B. Wang, et al., "Point cloud processing methods for 3D point cloud detection tasks," *ZTE Communications*, vol. 21, no. 4, pp. 38 – 46, Dec. 2023. doi: 10.12142/ZTECOM.202304005.

1 Introduction

3D object detection is critical for applications such as autonomous driving, robotic system navigation, and automation systems. The goal of 3D object detection is to locate and identify objects in 3D scenes. Specifically, its purpose is to estimate oriented 3D bounding boxes and semantic categories of objects from point cloud data and provide important information for subsequent analysis and processing. 3D point cloud object detection is a challenging task. Here are some major difficulties:

1) Occlusion: In complex scenes, target objects may be occluded by other objects, which affects the performance of detection algorithms.

2) Sparsity: Due to the working principle of light detection and ranging (LiDAR), point cloud data are usually sparse, which means that there are fewer effective points on the target object.

3) Point cloud noise: Noise points may be generated during the LiDAR scanning process, which will interfere with the performance of the detection algorithm.

4) Real-time requirements: 3D point cloud object detection usually needs to be completed in real time.

The object detection algorithm is a computer vision technology that can identify and locate specific objects in images or point clouds and is divided into 2D object detection[1 – 3] and 3D object detection[4 – 8]. These algorithms typically use deep learning techniques. Object detection includes tasks such as classification, localization, detection, and segmentation. Classification refers to obtaining what type of object is included in the image or point cloud data. Localization refers to the position of the given object. Detection refers to locating the position of an object and judging the category of the object. Segmentation refers to determining which object or scene each point or each pixel belongs to. Object detection algorithms are widely used in many fields, such as face recognition, automatic driving, and industrial inspection. For example, in face recognition, object detection algorithms can be used to automatically detect and track human faces and recognize the detected faces. Unmanned driving applications rely on object detection algorithms to give the poses of other traffic participants to deal with complex road conditions.

The point cloud processing method is a primary part of the

3D point cloud object detection algorithm. It can be roughly divided into the following two categories: the voxel-based point cloud processing method and point-based point cloud processing method.

The voxel processing method converts point cloud data into voxel representations, which are then processed using 3D convolutional neural networks (CNN). The point-based method is directly applied to the raw point cloud data, without converting it into grids. The point-based method can preserve the original characteristics and information of the point cloud and have lower computational cost and memory consumption.

However, the point-based method also faces some difficulties, such as dealing with the irregular structure and varying density of the point cloud and designing suitable algorithms or models for the points. Two major types of methods exist for processing the points: clustering-based methods[9–10] and deep learning-based methods[11–12]. Based on the clustering method, the appropriate clustering algorithm is selected and the clustering parameters are determined by the characteristics of the data. After denoising and merging adjacent clustering operations, the processing results are obtained. The method based on deep learning needs to prepare labeled data, select and train an appropriate deep learning model, and use the trained model to process the point cloud.

Our contribution can be summarized as follows:

1) We summarize voxel-based point cloud processing methods and find that the voxel-based methods can improve the processing performance of point clouds by optimizing the voxel partitioning scheme, improving the network structure for voxelized point clouds and the data structure for point clouds.

2) We summarize point-based point cloud processing methods and find that the point-based methods can improve the processing performance of point clouds by improving the sampling strategy of point clouds, combining the advantages of voxel-based methods, and optimizing the representation of points.

2 Basic Concepts and Metrics

1) Voxelization

Point cloud voxelization in Fig. 1 refers to the process of converting point cloud data into a voxel representation. Voxelization is to divide the point cloud into a spatially uniform voxel grid and generate many-to-one mapping between 3D points and their corresponding voxels.

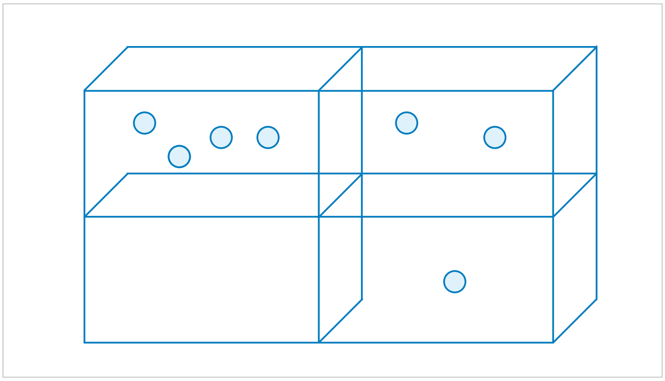
The voxelized point cloud data will be stored in memory in an orderly manner, which is beneficial to reduce random memory access and increase the efficiency of data calculation. Moreover, voxelization enables the ordered storage and downsampling of data, which allows such methods to handle much larger point cloud data. The voxelized data can also leverage spatial convolution effectively, which facilitates the extraction of local features at multiple scales and levels.

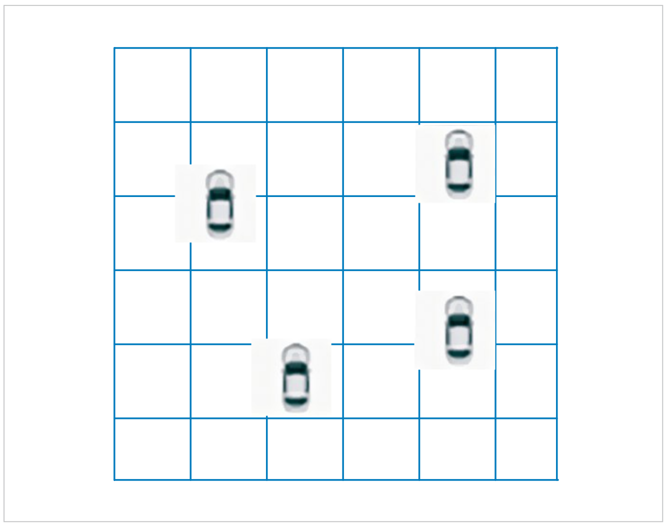
2) BEV

The bird's eye view (BEV) based algorithm is an advanced computer vision technique used in the field of autonomous driving. Using a combination of sensors and cameras, the algorithm creates a high-resolution overhead view of the vehicle's surroundings. The BEV perspective is shown in Fig. 2.

One of the advantages of BEV is that it provides a comprehensive view of the environment, providing a complete picture of the environment, unlike other computer vision techniques that only focus on specific objects in the environment. The perspective can provide more information for subsequent planning decisions. Another advantage is accuracy. A high-resolution top view can provide more accurate information. One disadvantage of the BEV-based algorithm is that BEV requires high computing power, which is challenging in real-time systems.

BEV is currently a very popular point cloud processing perspective. The methods related to BEV are proposed in Refs. [13–17]. Ref. [18] demonstrates the robustness capability of


▲Figure 1. Schematic representation of point cloud voxelization. Due to the sparsity and uneven distribution of point clouds, the number of point clouds in each voxel is unevenly distributed. There are even many voxels without point clouds. The voxel feature encoding (VFE) layer balances this through sampling


▲Figure 2. Bird's eye view based representation

the BEV method.

3) FPS

Farthest point sampling (FPS) is a commonly used sampling algorithm, especially suitable for LiDAR 3D point cloud data. It can guarantee uniform sampling of samples, so it is widely used. For example, in PointNet++[12], a 3D point cloud deep learning framework, sample points are sampled by FPS and then clustered as the receptive field; in VoteNet[19], the scattered points obtained by voting are sampled by FPS and then clustered; in PVN3D[20], a 6D pose estimation algorithm, it is used to select eight feature points of the object to vote and calculate the pose.

The principle of the FPS algorithm is: Given a point cloud with N points, a point P_0 is selected from the point cloud as the starting point to obtain a sampling point set $S = \{P_0\}$. Then we calculate the distance from all points to P_0 to form an N-dimensional array L, select the point corresponding to the maximum value as P_1, and update the sampling point set $S = \{P_0, P_1\}$. Then we calculate the distance from all points to P_1. For each point P_i, if the distance to P_1 is less than $L[i]$, $L[i] = \mathrm{d}(P_i, P_1)$ is updated. Therefore, the stored L in the array is always the shortest distance from each point to the sampling point set S. The point corresponding to the maximum value in L is then selected as P_2 and the sampling point set $S = \{P_0, P_1, P_2\}$ is updated. The above steps are repeated until N' target sampling points are sampled.

Several evaluation metrics are commonly used to assess the performance of an algorithm in 3D object detection. Here are some of the most common ones:

• Average precision (AP): This is a widely used metric that measures the accuracy of object detection algorithms. It is calculated by computing the area under the precision-recall curve. AP is often used to compare the performance of different algorithms on a given dataset.

• Intersection over union (IoU): This metric measures the overlap between the predicted bounding box and the ground truth bounding box. It is calculated as the ratio of the intersection area to the union area of the two boxes. IoU is often used as a threshold to determine whether a detection is true positive or false positive.

• Mean average precision (mAP): This metric is similar to AP, but it is calculated by taking the average of AP values across multiple object categories; mAP is often used to evaluate the overall performance of an object detection algorithm.

• Precision: This metric measures the proportion of true positives among all detections. It is calculated as TP/(TP + FP), where TP is the number of true positives and FP is the number of false positives.

• Recall: This metric measures the proportion of true positives among all ground truth objects. It is calculated as TP/(TP+FN), where TP is the number of true positives and FN is the number of false negatives.

These metrics are important for evaluating 3D object detection because they provide a quantitative measure of the object performance. By comparing these metrics across different algorithms, researchers can identify which ones are most effective for a given task.

3 Voxel-Based Point Cloud Processing Methods

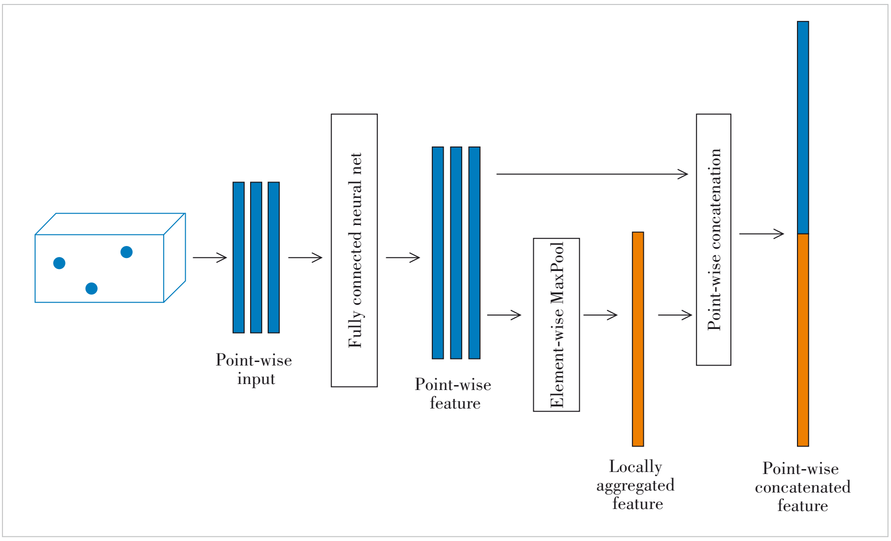
Voxel-based 3D point cloud object detection methods convert irregular point clouds into compact-shaped voxelized representations and then efficiently extract point cloud features for 3D object detection through 3D convolutional neural networks. During voxelization, the point cloud data are divided into a certain number of voxels, and these voxels are grouped and down-sampled. Since the point cloud data need to be down-sampled during the voxelization process, some detailed information will be lost. The degree of information loss is closely related to the chosen resolution.

Although the voxelization process causes information loss, it has many advantages. First, the voxelized point cloud data will be stored in an orderly manner in memory, which will help reduce random memory access and increase data computing efficiency. Second, thanks to the ordered storage and down-sampling of data brought about by voxelization, this type of method can handle point cloud data in a large amount. In addition, the voxelized data can efficiently be processed by spatial convolution, which is beneficial for extracting multi-scale and multi-level local feature information.

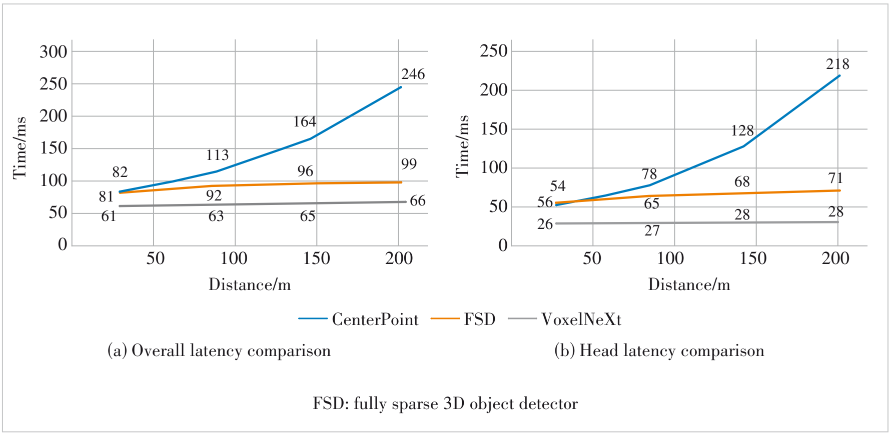
When it comes to voxel-based methods[5, 21 – 24], VoxelNet[21] has to be mentioned, which is a pioneering work. VoxelNet proposes a voxel feature encoding (VFE) layer, which groups points within a voxel in Fig. 1, and the number of point clouds after grouping is not exactly the same. In order to reduce the imbalance of the number of point clouds between groups, reduce the sampling deviation, and save computing resources, the grouped point clouds are randomly sampled so that the number of points in each group does not exceed a fixed value T. In each group, they apply PointNet[11] to learn features on each point and aggregate point features to obtain voxel-level features.

VFE is an important module. The VFE layer in Fig. 3 voxelizes the original 3D point cloud data and learns voxel-level features. This method combines the original point cloud representation and 3D voxel representation. After extracting features from the point cloud, VoxelNet uses convolutional middle layers and region proposal networks (RPNs)[25] to generate the final 3D detection box.

The key innovation of VoxelNeXt[5] is to omit the steps of anchor, sparse-to-dense, RPN, non max suppression (NMS), etc., and directly predict objects from sparse voxel features. Based on VoxelNet, VoxelNeXt has better accuracy and a speed trade-off than other detectors in nuScenes[26]. Compared with the CenterPoint[27], fully sparse 3D object detector (FSD)[28] and other methods, VoxelNeXt is more friendly to long-distance object detection in Fig. 4.


Point-wise input
Fully connected neural net
Point-wise feature
Element-wise MaxPool
Locally aggregated feature
Point-wise concatenation
Point-wise concatenated feature

▲**Figure 3. Each sampled voxel (the number of point clouds $t < T$) is transformed into a feature space point by point through a fully connected neural network and then the information is aggregated from the point features to encode the surface shape contained in the voxel. The aggregated features are obtained element by element through max pooling. The point-wise feature and locally aggregated feature connection are then aggregated to get a point-wise concatenated feature**



▲**Figure 4. Latency on Argoverse2 and various perception ranges**

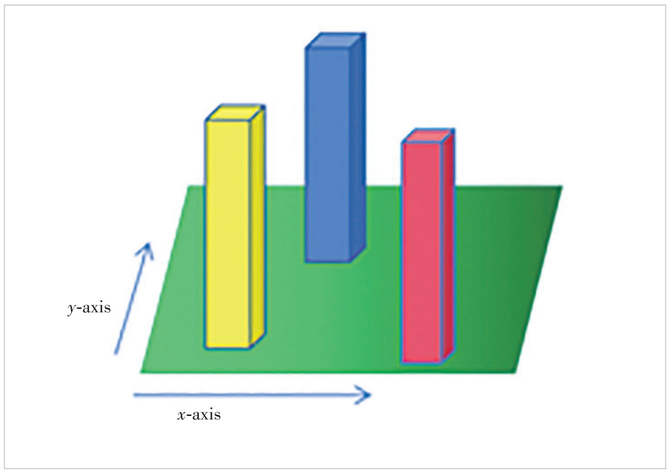
VoxelNeXt shows that fully sparse voxel-based representations are very effective for LiDAR 3D detection and tracking. VoxelNeXt proposed a fully sparse voxel-based network, which uses ordinary sparse convolutional networks for direct prediction. It uses only one extra down-sampling layer to optimize the sparse backbone network, and this simple modification enlarges the receptive field. This simple sparse linkage requires no additional parameterization layers and has a little additional computational cost.

VoxelNeXt places sparse features directly on the BEV plane and then combines features at the same location. It takes no more than 1 ms, but the effect is better than 3D sparse features. VoxelNeXt is entirely voxel-based and continuously clips irrelevant voxels along the down-sampling layer, which further saves computational resources and does not affect detection performance. Using the above-mentioned method to process voxels reduces calculation consumption without degrading performance.

The way of voxelization is not set in stone. For example, the classic VoxelNet[21] and sparsely embedded convolutional detection (SECOND)[22] divide the point cloud into a voxel to form a regular and dense voxel set, while SECOND uses sparse embedded convolution to improve efficiency.

To make a trade-off between accuracy and computation efficiency, PointPillars converts point clouds into pillars. Specifically, PointPillars divides the x axis and y axis of point cloud data into grids, and the data in the same grid is considered as a pillar (Fig. 5). This voxel division method can be considered to divide only one voxel on the z axis; P non-empty columns are generated after division; each column contains N point cloud data (more than N points are sampled as N points, and less than N points are filled with 0), and each point extracts D-dimensional features. There are nine features in PointPillar, which are $(x, y, z, r, x_c, y_c, x_p, y_p)$, where x, y and z are the 3D coordinates of the point, r is the reflection intensity, x_c and y_c are the distances from the center of the point cloud in the pillar, and x_p and y_p are the offset from the geometric center of the pillar.

In addition to improving the way of voxelization, the use of special data structures can also enhance the detection performance. The Octree-Based Transformer



▲**Figure 5. Pillar division scheme of PointPillars**

(OcTr) [29] algorithm first performs self-attention on the top level, constructs a dynamic octree on the hierarchical pyramid, and recursively propagates to the lower layer constrained by octants. This method can not only capture rich features from coarse-grained to fine-grained, but also control the computational complexity. Extensive experiments are conducted on Waymo Open Dataset[30] and Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) Dataset[31], and OcTr achieves new state-of-the-art results.

The performance of different detection models in three categories (Car, Pedestrian, and Cyclist) on the KITTI dataset[31] is listed in Table 1. The voxel-based method can achieve excellent performance by improving the processing method after extracting voxel features. Compared with VoxelNet, SECOND is modified to a sparse embedded convolution method to obtain a performance improvement. PointPillars proposes a way to balance accuracy and computational efficiency. A more appropriate voxel division method can achieve better results. The improvement made by OcTr is to use the transformer and the special octree data structure, which achieves better results. VoxelNeXt directly predicts objects based on sparse voxel features, without the need for sparse-to-dense conversion operations. In summary, voxelization is a method that can process large-scale point cloud data quickly and efficiently. The idea of voxelization is to process the unstructured point cloud into structured data and use the characteristics of CNN to process structured data to extract features from the point cloud. But nothing is perfect. Detailed information may be lost during the voxelization process, and voxelization is a computationally expensive step.

4 Point-Based Point Cloud Processing Methods

The point-based 3D point cloud object detection method is a method to perform object detection on the raw point cloud data. This approach preserves the unstructured form of the point cloud, but achieves a more compact representation by sampling the point cloud from its original size to smaller fixed-size N points. Sampling methods usually include random sampling and FPS, as well as several innovative sampling point methods[32].

Random sampling is achieved by randomly drawing points until N points are selected. But random sampling suffers from the scenario where points in denser regions of the point cloud are sampled more frequently than points in sparser regions of the point cloud. The FPS algorithm can mitigate this bias by using an iterative process to select points based on the furthest distance criterion. In each iteration, FPS first calculates the minimum distance from the unsampled point to the point set (the first point is randomly sampled and the second point is the point furthest from the first point) and then selects the furthest unsampled point .The final result is a more representative point cloud, but this method also suffers from expensive calculation costs.

The effect of PointNet[11] in point cloud-based methods is similar to that of VoxelNet in voxel-based methods. PointNet is a neural network-based approach that directly processes point cloud data for classification and segmentation. Operating directly on the raw point cloud eliminates unnecessary transformations of the data representation.

PointNet is a simple yet efficient point cloud feature extractor. It has three key modules: the symmetry function for unordered input, local and global information aggregation, and alignment network. Key to PointNet is that it can process unsorted point cloud data, when the disorder of point cloud is challenging in point cloud processing.

Based on Pointnet, Pointnet++[12] provides a hierarchical point cloud processing method that can effectively learn the local structure in the point cloud. Pointnet++ can handle more complex tasks such as scene segmentation, shape part segmentation, and 3D object detection.

The key technology of Pointnet++ is the introduction of hierarchical processing. Pointnet++ adopts a layered architecture. The entire point cloud is first sampled and then subdivided into smaller local areas and local features are learned on these local areas (set abstraction). Finally, these local features are aggregated to obtain global features. A Pointnet++ network consists of an encoder and a decoder. The encoder contains a collection abstraction module and the decoder contains a feature propagation module.

These two methods have promoted the application of point cloud data in the field of 3D vision and achieved remarkable research progress. There are also some extended methods[33–35]. Based on the PointNet series network, the feature extraction is directly applied to the original point cloud data.

▼Table 1. Performance of VoxelNet, SECOND, PointPillars and OcTr on the KITTI dataset[31]

Method	Modality	AP_{Car}			$AP_{Pedestrian}$			$AP_{Cyclist}$		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
VoxelNet[21]	LiDAR	81.97	65.46	62.85	57.86	53.42	48.87	67.17	47.65	45.11
SECOND[22]	LiDAR	83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	46.90
PointPillars[24]	LiDAR	79.05	74.99	68.30	52.08	43.53	41.49	75.78	59.07	52.92
OcTr[29]	LiDAR	88.43	78.57	77.16	61.49	57.17	52.35	85.29	70.44	66.17

AP: Average precision
KITTI: Karlsruhe Institute of Technology and Toyota Technological Institute
LiDAR: light detection and ranging
OcTr: Octree-Based Transformer
SECOND: sparsely embedded convolutional detection

This type of method is generally divided into two steps. The first step is often to propose a rough candidate frame and the second step is to adjust and refine the position of the candidate frame.

Down-sampling operations are generally required for point cloud processing. Down-sampling can not only reduce the amount of data, but also remove some noise to improve the quality of point cloud data to a certain extent.

Commonly used point cloud down-sampling methods include random sampling, uniform sampling, furthest point sampling, etc. Different sampling methods have different advantages. Random sampling has the lowest time complexity, but retains relatively few point cloud features. Uniform sampling can preserve the overall distribution of point clouds, but the disadvantage is that it retains fewer point cloud features and cannot retain more detailed information. The advantage of furthest point sampling is that it can retain edge information. It is suitable for large-scale data processing and can quickly complete down-sampling, but it has high time complexity.

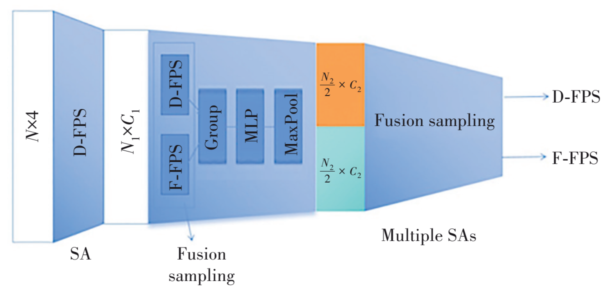
1) Improvement of the sampling method. The authors in Ref. [36] propose a lightweight and effective point-based 3D single stage object detector, named 3DSSD and believe that the feature propagation (FP) layer and refining process in the PointNet series methods[33, 37–38] will consume more than half of the time, but simply removing these modules and leaving only the set abstract (SA) layer to directly perform a single-stage proposal can result in a decrease in accuracy. They also believe that the down-sampling operation of the SA layer is based on the distance-based furthest point sampling method (D-FPS), which tends to retain background points. Therefore, they propose a new sampling method named F-FPS to filter background points and retain foreground points.

They use both spatial distance and semantic feature distance as the criterion in FPS. It is formulated as $C(A,B) = \lambda L_d(A,B) + L_f(A,B)$, where $L_d(A,B)$ is D-FPS, $L_f(A,B)$ is F-FPS, and λ is the balance factor. As shown in Table 2, F-FPS has the highest recall at $\lambda = 1.0$, where λ is the weight of D-FPS and F-FPS. Both the spatial distance and semantic feature distance are the criterion in FPS. In the experiment, 3DSSD adopts the method of fusion sampling. The points obtained by the two sampling methods each occupy half. The points are obtained after multi-layer SA as shown in Fig. 6. Then the candidate generation layer and two prediction heads predict the category and bounding box of the objects. 3DSSD greatly improves the speed of 3D object detection, and the speed exceeds 25 fps.

▼Table 2. Points recall among different sampling strategies on the nuScenes dataset. "4 096", "1 024" and "512" represent the number of representative points in the subset. The first row of results uses only D-FPS.

Method	Recall$_{4\,096}$	Recall$_{1\,024}$	Recall$_{512}$
D-FPS	99.7%	65.9%	51.8%
F-FPS, λ = 0.0	99.7%	83.5%	68.4%
F-FPS, λ = 0.5	99.7%	84.9%	74.9%
F-FPS, λ = 1.0	**99.7%**	**89.2%**	**76.1%**
F-FPS, λ = 2.0	99.7%	86.3%	73.7%

D-FPS: furthest point sampling based on 3D Euclidean distance
F-FPS: furthest point sampling based on feature distance



D-FPS: furthest point sampling based on 3D Euclidean distance
F-FPS: furthest point sampling based on feature distance
MLP: multi-layer perceptron
SA: set abstract

▲ **Figure 6. D-FPS is first used to down-sample the point cloud once. The point cloud is sampled, grouped, MLP and maximum pooled through the 1:1 combination of D-FPS and F-FPS sampling methods. The point cloud can be sampled multiple times in the same way**

2) Combination of point-based and voxel-based methods. There are some special methods that combine point-based and voxel-based methods[38–39]. Since the two methods have different advantages, their combination can bring more advantages.

The 3D object detector (STD)[38] has three main contributions. First, a spherical anchor is used to propose a point-based proposal generation example, which can achieve a high recall rate. Second, the point-based and voxel-based parts use the PointsPool link to predict efficiency and effectiveness, combining the advantages of VoxelNet[21] and PointNet[11]. Last, the alignment between classification scores and localization is achieved through a new 3D IoU prediction branch.

Point-voxel feature set abstraction for 3D object detection (PV-RCNN) [39] is a high-performance 3D object detection framework. It integrates the method of point-cloud voxelization and convolution and the method of PointNet-based set abstraction to obtain better point cloud features. PV-RCNN directly uses the original point cloud, processes the point cloud through 3D sparse convolution after voxelization and performs classification and box prediction through RPN on the BEV plane. At the same time, the FPS is used for key point sampling, and the key point features and the features of non-empty voxels around the key points are collected through the VSA module. These features are used to make up for the information loss during voxelization. Object category and bounding box predictions are refined through a two-part combination.

Both PV-RCNN and STD have achieved good results on the KITTI dataset (Table 3), and their performance outperforms either the voxel-based or point-cloud-based method used alone,

▼**Table 3. Performance testing on the KITTI test set. Mean average precision is taken as the evaluation metric. The table shows better performance of PV-RCNN and STD**

Method	$AP_{\text{Car-3D Detection}}$			$AP_{\text{Car-BEV Detection}}$			$AP_{\text{Cyclist-3D Detection}}$			$AP_{\text{Cyclist-BEV Detection}}$		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
SECOND[22]	83.34	72.55	65.82	89.39	83.77	78.59	71.33	52.08	45.83	76.50	56.05	49.45
Fast Point R-CNN[35]	85.29	77.40	70.24	90.87	87.84	80.52	-	-	-	-	-	-
STD[38]	87.95	79.71	75.09	94.74	89.19	**86.42**	78.69	61.59	55.30	81.36	67.23	59.35
PV-RCNN[39]	**90.25**	**81.43**	**76.82**	**94.98**	**90.65**	86.14	78.60	**63.71**	57.65	82.49	**68.89**	**62.41**

AP: average precision
BEV: bird's eye view
PV-RCNN: point-voxel feature set abstraction for 3D object detection
R-CNN: Region-CNN
SECOND: Sparsely Embedded Convolutional Detection
STD: Sparse-to-Dense 3D Object Detector for Point Cloud

demonstrating the benefits of combining these two complementary methods.

Besides STD and PV-RCNN, the methods proposed in Refs. [40] and [41] also combine the point-based and voxel-based manners.

Some methods use other networks such as graph neural networks (GNNs). They convert the point cloud into a regular grid or voxel and use CNN (point cloud representation in the grid) or deep learning technology to process the point cloud (point cloud in the point set) after obtaining the point set through sampling and other operations (Fig. 7). In addition, Point-GNN[42] constructs the point cloud into a graph. It has three main components: graph construction from point cloud, graph neural network for object detection, and bounding box merging and scoring. Specifically, the points in the point cloud are used as N vertices, with a point as the center and r as the radius. Neighboring points in the range are concatenated to construct a graph $G = (P, E)$, for example:

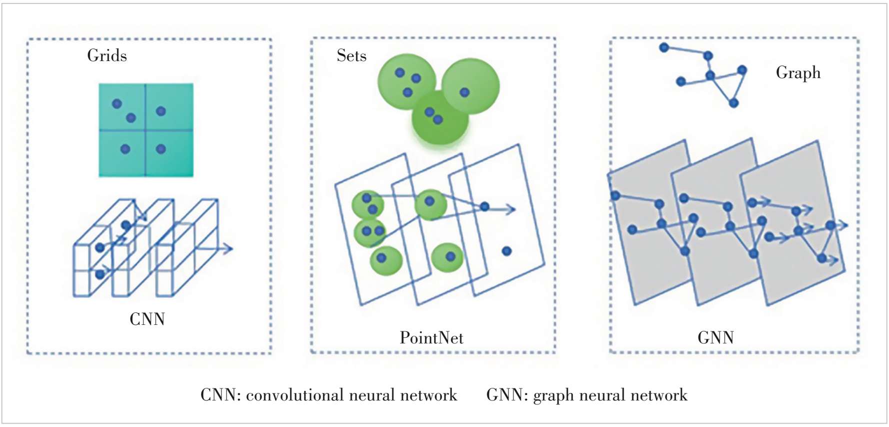
$$E = \left\{ \left(p_i, p_j \right) \middle| \left| x_i - x_j \right|^2 < r \right\}.$$

In order to reduce complexity, Point-GNN uses voxel operations to down-sample point clouds in the actual process and the voxels are only used for reducing the point cloud density. Once constructed, the point cloud is processed using a multi-iteration GNN[43].

Point-GNN has achieved excellent performance on the KITTI test data set. The average precision of the car, pedestrian and cyclist at the easy level reached 88.33, 51.92 and 78.60, respectively, at the modality levels 79.47, 43.77 and 63.48, respectively, and at the hard level 72.29, 40.14 and 57.08, respectively. The detection performance of the car and cyclist surpasses both the radar-only methods such as STD[38] and PointRCNN[33] and the radar and image fusion methods such as AVOD-FPN[44] and UberATG-MMF[45].

The point-based methods still have several modules that need to be improved. One module is sampling, which can reduce the consumption of computing resources by selecting a subset of points from the original point cloud. However, sampling may cause some information loss, which affects the quality of the features that can be extracted in subsequent operations. Therefore, the choice of the sampling algorithm is crucial for the point-based method. For example, PointNet++ uses feature propagation to suppress the information loss caused by sampling, and 3DSSD improves different sampling methods to retain more useful information and improve efficiency.

Another module that can be improved is voxelization, which is a special method to introduce voxels into point cloud processing. Voxels are small cubes that divide the three-dimensional space and contain a certain number of points. The advantage of voxelization is to convert point clouds into ordered data and also reduce computational complexity. However, voxelization may introduce quantization errors and lose some fine-grained details. Therefore, some methods combine the information obtained from both voxels and points to improve performance. For example, PV-RCNN uses voxel-based RPN and point-based RoI feature extractors (RoIFEs) to achieve state-of-the-art results on 3D object detection.



▲**Figure 7. Representation of point-cloud grids, sets and graph and their corresponding processing methods**

The third module that can be improved

is the basic network model, which is used to process the point cloud data and extract features. Different network models have different advantages and disadvantages for point cloud processing. For example, GNN can capture the structure and relationship of point cloud data by using nodes and edges. It can handle irregular and unordered data better than convolutional neural networks.

5 Conclusions

In this paper, we summarize the processing of point clouds in object detection. Point cloud processing is the first step in most models and it can greatly affect the performance of subsequent detection operations. Point cloud processing can be divided into two categories: voxel-based and point-based processing, both of which have their own advantages and disadvantages.

Voxel-based processing is a method that divides the three-dimensional space into small cubes called voxels and assigns points to voxels according to their coordinates. The advantage of voxel-based processing is that it can convert point clouds into ordered data and reduce computational complexity. However, voxel-based processing may introduce quantization errors and lose some fine-grained details. Many works have improved voxel-based methods by changing the way voxels are divided, changing the network for processing voxels, changing the data structure for processing data, etc. These approaches can reduce time complexity and organize voxel-level features well, further improving performance.

Point-based processing is a method that directly operates on raw points without any transformation or quantization. The advantage of point-based processing is that it can preserve the original structure and information of point clouds. However, point-based processing may face challenges such as irregularity and sparsity of point clouds. Many works have improved point-based methods by improving the way of point cloud sampling, introducing some voxel-based features or directly obtaining the graph structure from the structure of the original point cloud data. These approaches can enhance the feature extraction and representation of points, which can also significantly improve the performance of subsequent detection.

References

[1] REN S Q, HE K M, GIRSHICK R, et al. Faster R-CNN: towards real-time object detection with region proposal networks [J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 39(6): 1137 – 1149. DOI: 10.1109/tpami.2016.2577031

[2] REDMON J, DIVVALA S, GIRSHICK R, et al. You only look once: unified, real-time object detection [EB/OL]. (2016-05-09)[2023-08-20]. https://arxiv.org/abs/1506.02640

[3] LAW H, DENG J. CornerNet: detecting objects as paired keypoints [J]. International journal of computer vision, 2020, 128(3): 642 – 656. DOI: 10.1007/s11263-019-01204-1

[4] SHI S S, WANG X G, LI H S. PointRCNN: 3D object proposal generation and detection from point cloud [EB/OL]. (2019-05-16)[2023-08-21]. https://arxiv.org/abs/1812.04244

[5] CHEN Y K, LIU J H, ZHANG X Y, et al. VoxelNeXt: fully sparse VoxelNet for 3D object detection and tracking [EB/OL]. (203-03-20)[2023-08-21]. https://arxiv.org/abs/2303.11301

[6] YANG Z T, SUN Y N, LIU S, et al. STD: sparse-to-dense 3D object detector for point cloud [EB/OL]. (2019-07-22) [2023-08-21]. https://arxiv. org/abs/1907.10471

[7] PHILION J, FIDLER S. Lift, splat, shoot: encoding images from arbitrary camera rigs by implicitly unprojecting to 3D [C]//European Conference on Computer Vision. Springer, 2020: 194 – 210.10.1007/978-3-030-58568-6_12

[8] YIN T W, ZHOU X Y, KRÄHENBÜHL P. Center-based 3D object detection and tracking [C]//2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2021: 11779 – 11788. DOI: 10.1109/CVPR46437.2021.01161

[9] KLASING K, WOLLHERR D, BUSS M. A clustering method for efficient segmentation of 3D laser data [C]//2008 IEEE International Conference on Robotics and Automation. IEEE, 2008: 4043 – 4048. DOI: 10.1109/ROBOT.2008.4543832

[10] KLASING K, WOLLHERR D, BUSS M. Realtime segmentation of range data using continuous nearest neighbors [C]//2009 IEEE International Conference on Robotics and Automation. IEEE, 2009: 2431 – 2436. DOI: 10.1109/ROBOT.2009.5152498

[11] CHARLES R Q, HAO S, MO K C, et al. PointNet: deep learning on point sets for 3D classification and segmentation [C]//2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2017: 77 – 85. DOI: 10.1109/CVPR.2017.16

[12] QI C R, YI L, SU H, et al. PointNet++: deep hierarchical feature learning on point sets in a metric space [EB/OL]. (2017-06-07)[2020-08-21]. https://arxiv.org/abs/1706.02413

[13] HUANG J J, HUANG G, ZHU Z, et al. BEVDet: high-performance multi-camera 3D object detection in bird-eye-view [EB/OL]. (2022-06-16)[2023-08-21]. https://arxiv.org/abs/2112.11790

[14] LI Z Q, WANG W H, LI H Y, et al. Bevformer: learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers [EB/OL]. (2022-07-13)[2023-08-21]. https://arxiv.org/abs/2203.17270

[15] LI Y H, GE Z, YU G Y, et al. BEVDepth: acquisition of reliable depth for multi-view 3D object detection [EB/OL]. (2022-11-30)[2023-08-21]. https://arxiv.org/abs/2206.10092

[16] LIU Z J, TANG H T, AMINI A, et al. BEVFusion: Multi-task multi-sensor fusion with unified bird's-eye view representation [EB/OL]. (2022-06-16)[2023-08-21]. https://arxiv.org/abs/2205.13542

[17] WANG R H, QIN J, LI K Y, et al. BEV-LaneDet: a simple and effective 3D lane detection baseline [EB/OL]. (203-03-11)[2023-08-21]. https://arxiv. org/abs/2210.06006

[18] DONG Y P, KANG C X, ZHANG J L. Benchmarking robustness of 3D object detection to common corruptions in autonomous driving [EB/OL]. (2023-03-20)[2023-08-21]. https://arxiv.org/abs/2303.11040

[19] QI C R, LITANY O, HE K M, et al. Deep hough voting for 3D object detection in point clouds [EB/OL]. (2019-08-22) [2023-08-21]. https://arxiv. org/abs/1904.09664

[20] HE Y S, SUN W, HUANG H B, et al. PVN3D: deep point-wisea 3D keypoints voting network for 6DoF pose estimation [EB/OL]. (2020-03-24)[2023-08-21]. https://arxiv.org/abs/1911.04231

[21] ZHOU Y, TUZEL O. VoxelNet: end-to-end learning for point cloud based 3D object detection [C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE, 2018: 4490 – 4499. DOI: 10.1109/CVPR.2018.00472

[22] YAN Y, MAO Y X, LI B. SECOND: sparsely embedded convolutional detection [J]. Sensors, 2018, 18(10): 3337. DOI: 10.3390/s18103337

[23] SIMON M, AMENDE K, KRAUS A, et al. Complexer-YOLO: real-time 3D object detection and tracking on semantic point clouds [EB/OL]. (2019-04-16)[2023-08-21]. https://arxiv.org/abs/1904.07537

[24] LANG A H, VORA S, CAESAR H, et al. PointPillars: fast encoders for object detection from point clouds [EB/OL]. (2020-03-24)[2023-08-21]. https://arxiv.org/abs/1812.05784

[25] REN S Q, HE K M, GIRSHICK R, et al. Faster R-CNN: towards real-time ob-

ject detection with region proposal networks [J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 39(6): 1137 – 1149. DOI: 10.1109/TPAMI.2016.2577031

[26] CAESAR H, BANKITI V, LANG A H, et al. nuScenes: a multimodal dataset for autonomous driving [EB/OL]. (2020-05-05)[2023-08-21]. https://arxiv.org/abs/1903.11027

[27] YIN T W, ZHOU X Y, KRÄHENBÜHL P. Center-based 3D object detection and tracking [EB/OL]. (2021-01-06) [2023-08-21]. https://arxiv. org/abs/2006.11275

[28] FAN L, WANG F, WANG N Y, et al. Fully sparse 3D object detection [EB/OL]. (2022-10-03)[2023-08-21]. https://arxiv.org/abs/2207.10035

[29] ZHOU C, ZHANG Y N, CHEN J X, et al. OcTr: octree-based transformer for 3D object detection [EB/OL]. (2023-03-22)[2023-08-21]. https://arxiv.org/pdf/2303.12621.pdf

[30] SUN P, KRETZSCHMAR H, DOTIWALLA X, et al. Scalability in perception for autonomous driving: waymo open dataset [EB/OL]. (2023-03-22)[2023-08-21]. https://arxiv.org/abs/1912.04838

[31] GEIGER A, LENZ P, URTASUN R, et al. Are we ready for autonomous driving? The KITTI vision benchmark suite [C]//2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2012. DOI: 10.1109/CVPR.2012.6248074

[32] YANG Z T, SUN Y N, LIU S, et al. 3DSSD: point-based 3D single stage object detector [EB/OL]. (2020-02-24)[2023-08-21]. https://arxiv.org/abs/2002.10187

[33] SHI S S, WANG X G, LI H S. PointRCNN: 3D object proposal generation and detection from point cloud [C]//2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020: 770 – 779. DOI: 10.1109/CVPR.2019.00086

[34] QI C R, LIU W, WU C X, et al. Frustum PointNets for 3D object detection from RGB-D data [EB/OL]. (2018-04-13)[2023-08-21]. https://arxiv. org/abs/1711.08488

[35] CHEN Y L, LIU S, SHEN X Y, et al. Fast point R-CNN [EB/OL]. (2019-08-16)[2023-08-21]. https://arxiv.org/abs/1908.02990

[36] YANG Z T, SUN Y N, LIU S, et al. 3DSSD: point-based 3D single stage object detector [C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020: 11037 – 11045. DOI: 10.1109/CVPR42600.2020.01105

[37] YANG Z T, SUN Y N, LIU S, et al. IPOD: intensive point-based object detector for point cloud [EB/OL]. (2018-12-13)[2023-08-21]. https://arxiv. org/abs/1812.05276

[38] YANG Z T, SUN Y N, LIU S, et al. STD: sparse-to-dense 3D object detector for point cloud [C]//2019 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, 2020: 1951 – 1960. DOI: 10.1109/ICCV.2019.00204

[39] SHI S S, GUO C X, JIANG L, et al. PV-RCNN: point-voxel feature set abstraction for 3D object detection [EB/OL]. (2021-04-09)[2023-08-21]. https://arxiv. org/abs/1912.13192

[40] YOU Y R, WANG Y, CHAO W-L, et al. Pseudo-lidar++: accurate depth for 3d object detection in autonomous driving [EB/OL]. (2020-02-15)[2023-08-21]. https://arxiv.org/abs/1906.06310

[41] HE C H, ZENG H, HUANG J Q, et al. Structure aware single-stage 3D object detection from point cloud [C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020: 11870 – 11879. DOI: 10.1109/CVPR42600.2020.01189

[42] SHI W J. Point-GNN: graph neural network for 3D object detection in a point cloud [EB/OL]. (2020-03-02)[2023-08-21]. https://arxiv.org/abs/2003.01251

[43] SCARSELLI F, GORI M, TSOI A C, et al. The graph neural network model [J]. IEEE transactions on neural networks, 2009, 20(1): 61 – 80. DOI: 10.1109/tnn.2008.2005605

[44] KU J, MOZIFIAN M, LEE J, et al. Joint 3D proposal generation and object detection from view aggregation [C]//2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). ACM, 2018: 1 – 8. DOI: 10.1109/IROS.2018.8594049

[45] LIANG M, YANG B, CHEN Y, et al. Multi-task multi-sensor fusion for 3D object detection [C]//2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020: 7337 – 7345. DOI: 10.1109/CVPR.2019.00752

Biographies

WANG Chongchong received his BS degree in computer science and technology from Huazhong Agricultural University, China in 2022. He is currently pursuing a master's degree in computer science and technology at Anhui University, China.

LI Yao received his BS degree in electronic information engineering from Jilin University, China in 2019. He is currently pursuing a PhD degree in computer science and technology at the University of Science and Technology of China. His research interests include computer vision and intelligent transportation perception system.

WANG Beibei graduated from University of Science and Technology of China with BS in physics in 2014. He furthered his studies at the University of Southern California, USA, where he obtained PhD in physics in 2020 and MS in computer science in parallel. His research interests currently focus on computer vision and multimodal perception methods for autonomous systems.

CAO Hong received his PhD degree from Zhejiang University, China in 2014. He is currently a associate research fellow with the Institute of Artificial Intelligence, Hefei Comprehensive National Science Center (Anhui Artificial Intelligence Laboratory), China. His research interests include autonomous driving, roadside perception and robotics.

ZHANG Yanyong (yanyongz@ustc.edu.cn) received her BS from the University of Science and Technology of China (USTC) in 1997, and PhD from Penn State University in 2002. From 2002 and 2018, she was on the faculty of the Electrical and Computer Engineering Department at Rutgers University, USA. She was also a member of the Wireless Information Networks Laboratory (Winlab). Since July 2018, she joined the school of Computer Science and Technology at USTC. She has 21 years of research experience in the areas of sensor networks, ubiquitous computing, and high-performance computing, and has published more than 140 technical papers in these fields. She received the NSF CAREER award in 2006, and was elevated to IEEE Fellow in 2017. She has served/currently serves as the Associate Editor for several journals, including *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Service Computing*, *IEEE Transactions on Dependable and Secure Computing*, and *Elsevier Smart Health*. She has served on various conference TPCs including DSN, Sensys, Infocom, etc. She is the TPC co-chair of IPSN'22.