The Journal of Systems & Software 204 (2023) 111794

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Effective software security enhancement using an improved PointNet++[☆]

Yi Zhao, Hao Chen, Liang Zen, Zhao Li [*]

College of Mathematics and Computer Science, Guangdong Ocean University, Zhanjiang, China

A R T I C L E I N F O

Article history:
Received 25 January 2023
Received in revised form 11 May 2023
Accepted 25 June 2023
Available online 28 June 2023

Keywords:
Driving software security
PointNet++
Density related-farthest point sampling
Farthest point sampling
3D point cloud

A B S T R A C T

As common three-dimensional (3D) data, point clouds have wide application prospects in many fields. The point cloud disorder problem is solved by the PointNet neural network using a symmetric function, which insensitivity to the input order enables PointNet to process the original point cloud data directly. The ability to extract local features was enhanced by introducing the PointNet++, furnishing a better solving capacity of 3D vision problems and improved intelligent driving software security. This paper analyzes the PointNet++ implementation principles and improves its local feature extraction capability by the proposed density-related farthest point sampling (DR-FPS) algorithm, mitigating some limitations of the conventional FPS algorithm so that the sampling results can better express the feature information of point cloud data. The accuracy of the proposed DR-FPS algorithm applied to five-category and ten-category classification tasks exceeded that of the conventional FPS by 4.4 and 5.6%, respectively. Finally, a positive correlation between the accuracy increment and the number of classification categories was revealed. The results are instrumental to intelligent driving software security enhancement.

1. Introduction

In recent years, driven by smaller and cheaper laser scanning technology armed with improvements in computer vision, an increasing number of scholars have paid attention to 3D vision based on point clouds (Wang et al., 2023; Guo et al., 2020a; Li et al., 2021; Mahendru and Kumar, 2022). Although two-dimensional image data are relatively easy to obtain, many ready-made datasets are available for research on the Internet. In the field of object recognition, it relies more on two-dimensional image data. Two-dimensional image data also have shortcomings. For example, it cannot measure the shape, is easily affected by lighting conditions, and is sensitive to object motion. For these reasons, two-dimensional data have certain limitations in tasks involving spatial information (Shi et al., 2017; Wei et al., 2023). Unlike 2D image data, 3D point cloud technology has rich spatial information and higher accuracy. Application of smaller and cheaper scanners dropped the cost of obtaining point cloud data and increased the number of 3D point cloud datasets, such as ModelNet, ShapeNet (Chang et al., 2015), and S3DIS, promoting the 3D vision development.

With the increasing demand for accuracy and automation, 3D point cloud technology has become a research hotspot in unmanned driving, remote sensing, medical care, and other fields, with broad application prospects. Point cloud data refer to a set of vectors in a three-dimensional coordinate system. The data obtained by scanning are recorded in the form of points. Each point contains three-dimensional coordinates; some may contain color information (RGB) or reflection intensity information (Intensity). The point cloud data are the data collected by lidar. Because of the noncontact measurement characteristics of laser radar, it has the advantages of fast measurement speed, high accuracy, accurate recognition, and so on. It has become the core sensor for the positioning and navigation of mobile robots. In the field of laser radar technology, at present, triangulation (Hartley and Sturm, 1995) and time-of-flight (TOF) (Tanaka et al., 1988) methods are mainly used to measure the distance. The triangulation method is mainly used to solve the triangle through the spot imaging position of the camera. In recent years, driven by the development of computer vision, driverless technology has developed very rapidly. The logic of realizing driverless driving is to recognize the environmental state around the vehicle through various sensors and analyze it by themselves. The key to autonomous control action is to perceive the surrounding information, and point clouds play an important role in sensing the environment because of their rich location information (He et al., 2021). Accurately identifying each instance of a 3D point cloud is the core of intelligent driving software.

In remote sensing, the detection and mapping of geological landforms play an important role in predicting address disasters, analyzing hidden dangers of disasters, and analyzing geographical

[☆] Editor: Prof W. Eric Wong.
[*] Corresponding author.
E-mail address: ivwepriu@sina.com (Z. Li).

https://doi.org/10.1016/j.jss.2023.111794
0164-1212/© 2023 Elsevier Inc. All rights reserved.

features. It is usually necessary to complete the classification and analysis of vegetation, water, rocks, and other objects. The surface information of these objects is very rich. Still, the information carried by conventional two-dimensional pictures is extremely limited due to the limitation of pictorial representation. Two-dimensional pictures can only analyze objects in a limited dimension, and the resulting information obtained through analysis is also relatively limited. The 3D contour of the object and the spatial coordinates of each point on the surface can be accurately obtained by scanning the address geomorphology by lidar. The scanned 3D point cloud also has stronger information expression ability and has been widely used in geological geomorphology exploration and mapping (Rieg et al., 2014).

In the military field, the precise guided attack of missiles requires identifying the identity and key information of the target at any angle to implement the precise attack (Deng et al., 2020). It can only express part of the target information from a two-dimensional perspective through satellite remote sensing and aerial reconnaissance images. In limited circumstances, the geometric information of the target provided is also very limited and affected by the brightness of the target. It is not easy to accurately analyze the depth information of the target in a two-dimensional (2D) image, which greatly impacts the system's guidance accuracy. The 3D point cloud data scanned by lidar can accurately express the geometric information of the object, which overcomes the defect that the two-dimensional image is affected by the environment and object state to a great extent, and it has great significance and far-reaching impact on military development and space strategic security.

In the field of ocean exploration, point cloud technology is also widely used. Underwater robots are an important method of seabed exploration. It mainly receives and collects underwater acoustic information through sonar to realize the collection and identification of underwater object information. As proven by a large amount of surveying and mapping data, lidar has better effects and characteristics than sonar regarding collection accuracy and information delay lag. The development of 3D point cloud neural networks has achieved higher accuracy and segmentation results in processing point cloud data. The development of 3D neural networks can provide a more efficient method of ocean exploration and seabed terrain rendering and help the development of related disciplines (Tang et al., 2020).

In the medical field, 3D points cloud also have many applications. With the development of technology, high-precision medical facilities have also been applied to clinical medicine, including medical robots (Mavroidis et al., 1998). For example, a laser acupuncture robot is a medical robot with a mechanical arm as the carrier and a laser acupuncture therapeutic instrument as the actuator (Tao et al., 2010). The key in the treatment process is to find the corresponding acupoints and dynamically plan the posture of the mechanical arm during the treatment. Because the object is moving during the treatment process, using three-dimensional point cloud vision technology can better identify the corresponding acupoints with higher accuracy. Because of the high requirements for accuracy in the medical field, point cloud technology has considerable room for development.

3D point cloud technology can accurately identify foreign objects on railways. Foreign object detection in important areas such as station platforms and tunnel entrances can effectively ensure the safety of high-speed railway operation, and lidar is not vulnerable to weather and the environment. 3D point cloud semantic segmentation technology can segment the point cloud data of road environments, locate and map simultaneously, and identify travelers, cars, and other objects, which is instrumental in autonomous driving vehicle software security.

Our preliminary study of the PointNet++ neural network, with the deep hierarchical feature learning on point sets in a metric space, revealed that although the farthest point sampling (FPS) algorithm covered the whole dataset quite comprehensively (Qi et al., 2017b), the selected points had a small or even zero number of points within their given radius, deteriorating the feature extraction capacity. This paper proposes density-related farthest point sampling (DR-FPS) to solve this problem by improving the FPS algorithm in PointNet++ and verifying it on the five- and ten-class classification tasks (Qi et al., 2017a). The experimental results show that compared with the original algorithm, it improves the accuracy by 4.4 and 5.6%, respectively.

2. Related work

A point cloud refers to a massive set of points that express targets' spatial distribution and surface characteristics in the same spatial reference system. Each point contains *X*, *Y*, and *Z* geometric coordinates, intensity values, classification values, etc. The combination of these points is the point cloud. Point clouds can truly restore the 3D features of the target, and they are an important tool for visualizing objects.

Because of the permutation invariance and rigid body motion invariance of point cloud data, deep neural networks, and other algorithms cannot directly process point cloud data. They can only be processed by converting point cloud data into "nonpoint cloud data" after certain processing. Still, the processed data will usually become unnecessarily large, and some geometric information of point cloud data will be lost. Therefore, how to apply point cloud data directly to deep learning networks has become a problem for scholars.

Compared with conventional algorithms, deep learning does not need to design features manually, and it can optimize the loss function to learn rules as much as possible. With the successful application of two-dimensional images in deep learning, people began to study the combination of three-dimensional data and deep learning. For the application of point clouds in deep learning, we usually divide point cloud classification methods into the following three ways according to the data types input by neural networks: Multiview-based, Volume-based (Guo et al., 2020b), and Point-based methods.

2.1. *Implementation based on the multi-view method*

The MVCNN proposed by Su et al. (2015). reduced the dimensionality of 3D images and then used a convolutional neural network to process them. Its basic logic was to use the rendering engine to generate a batch of 2D images from multiple perspectives of the 3D data and input these 2D images into the MVCNN network as original training data. In MVCNN, the two-dimensional images from each perspective first go through the CNN1 convolution network once, then go through the "grouping layer" once in the view pooling layer, and then are input to the CNN2 convolution network to obtain results.

Because the view pooling layer in MVCNN cannot distinguish the differences between each view, Feng et al. proposed GVCNN [18]. Similar to MVCNN, it is based on multi-view in processing 3D images. Still, it introduces the Grouping Module to study the similarity and differences between views, and the accuracy is 2%–3% higher than that of MVCNN.

2.2. *Volume-based implementation*

Volume-based methods transform point cloud data into 3D voxels of pixels similar to 2D images by calculating the difference between the maximum and minimum values of the input point cloud data in the three directions (*X*, *Y*, and *Z*) of 3D coordinates. These denote height, width, and depth, a cube is determined to surround the point cloud, and then the basic voxels are divided into small voxels. The more voxels that are divided, the higher the resolution will be. Finally, calculate the inner centroid of nonempty voxels to complete the down sampling and realize the voxelization of point clouds.

The representative voxelization method for point cloud data processing is VoxNet, which was first proposed in the literature (Feng et al., 2018). Its network structure is simple. The value in a voxel is obtained through the occupancy grid layer and then enters the CNN. Its principle is to express the environmental state as a 3D grid of random variables through the occupancy grid and estimate the probability of maintaining its occupancy rate according to the incoming sensor data and prior knowledge. This shows that CNN's ability to deal with voxels occupying the grid is very excellent.

Although vowelized point cloud data can be better applied to deep learning, some data will inevitably be lost in its transformation. The space storage overhead and computing time overhead are large (Maturana and Scherer, 2015).

2.3. Point-based implementation

Point cloud data could not be directly processed for a long time until Qi et al. proposed PointNet (Liu et al., 2020), which formally solved this problem.

PointNet designs a new deep net structure suitable for 3D unordered point sets, which can directly process point cloud data and solves the problems of permutation invariance and transformation invariance of point clouds. At the same time, PointNet shows better speed and performance than other neural networks in classification, component segmentation, and scene segmentation tasks. To solve the disorder of the point cloud, PointNet processes each point separately and uses the symmetry function (maximum pooling) to make the result unique regardless of the order in which the points in the point cloud are arranged. In PointNet, each point is transformed separately. Hence, the input format is easy to carry out the rigid or affine transformation. PointNet converts the original point matrix or characteristic matrix into standard orientation by adding a T-Net to adapt to point clouds with different orientations and ensure the change invariance of point clouds.

Although PointNet can directly use point cloud data for training, it lacks feature extraction layer by layer. It only represents each point, which is too weak to integrate local structural information. To solve these problems, the authors of PointNet used the idea of CNN for reference, first dividing the point cloud into local regions with overlapping parts, then using PointNet to extract local features in the local regions, then expanding the scope, and then extracting higher-level features from the scope of local features until finally extracting the global features of the point cloud. Based on this principle, PointNet++ is proposed.

PointNet++ uses the penalty model to define the local area for local division. In all datasets, several points are selected as the center points of the ball, and the designed radius is used to draw a circle to cover all datasets. In selecting the central point, the farthest point sampling algorithm is adopted. A point is randomly selected, and then the point farthest from this point is selected to join the result set. The process is iterated until the number of points in the result set reaches a given value, which is also the core of PointNet++ to obtain local features.

Although the farthest point sampling algorithm can comprehensively cover a complete dataset, it is only possible to judge the importance of the data by taking the distance information as the standard. In selecting such points, there will inevitably be a small number of nearby points. Based on this, we propose (DR-FPS). The idea of DR-FPS is to add a density filter to the farthest point algorithm. The higher the density, the richer the information contained. By sampling the farthest points related to point density, regions with more abundant local information are selected. Such points can cover the whole dataset more completely and consider that the divided regions have more abundant information.

Table 1
FPS algorithm steps.

```
Algorithm 1: FPS algorithm
Input: $L$, point cloud data $x\ y\ z$; number of sampling points
Output: sampling point set $S$
1 $L$ = int[$xzy.length$]*$1e$10// Distance matrix L between all points and point set $S$
2   $S$.append($P_0$)// Randomly select the initial sampling point $P_0$ and add it to the point set $S$
3 For each $i$ in range($npoint$)// Calculate the distance from other points to the current sampling point
4 $dist$ = distance ($XYZ$, $P_i$)
5 $L$ = min($L$, $dist$)// Take the minimum value to update the matrix $L$
6 $P_i$ = max($L$)//Pick the value with the largest distance
7 $S$.append($P_i$)//add to point set $S$
8 return $n$
```

2.4. Farthest point sampling (FPS) algorithm

The farthest point sampling (FPS) algorithm is used in the sampling layer of the PointNet++ neural network set abstraction module. The farthest point sampling algorithm is one of the more common down sampling algorithms in 3D point clouds. This algorithm selects some points from the point set data of the point cloud to maximize the distance between these points. In the process of random sampling, the selection probability of all points is the same, which easily leads to some feature points containing key information in the original point cloud data not being selected, some outliers will be selected, and the selected points will be unevenly distributed, resulting in the loss of key information of objects in the sampling process and the concentration of sampling points, resulting in unclear contour features. Compared with random sampling, FPS can generate a group of evenly distributed points on the surface of the point cloud. The sampling result point set also has a more obvious and prominent contour in three-dimensional space, showing better sampling performance than the random sampling method.

The main implementation steps of the FPS algorithm are listed in Table 1.

The FPS algorithm solves the problem of uneven point selection in 3D point cloud processing data. The algorithm can select a group of representative points depicting the shape of the object more evenly and represent the general information of the point cloud with fewer representative points, which greatly reduces the amount of calculation of the neural network model for point cloud data processing and greatly reduces the pressure in the data processing process and the time cost of neural network model training and testing. The point cloud model abstracted by the FPS algorithm has little impact on the processing results of the neural network.

3. Materials and methods

This paper adopts the PointNet++ neural network (Zhang and Lin, 2016), which can be subdivided into the following three layers: the Set Abstraction layer, the fully connected layer, and class scores, as shown in Fig. 1.

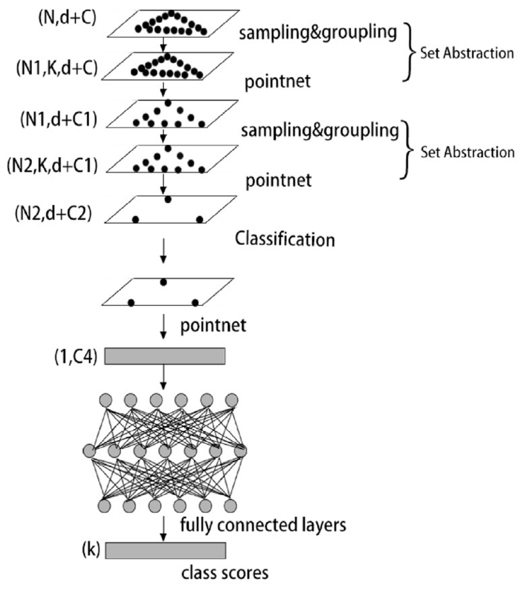


Fig. 1. The PointNet++ framework for 3D shape classification.

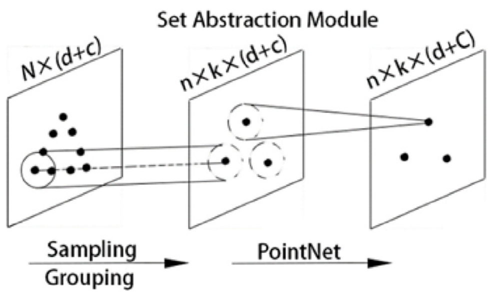


Fig. 2. Set abstraction layer.

3.1. Set abstraction layer

As a concrete implementation of multi-level structure feature extraction, the set abstraction layer is the core of the entire PointNet++ neural network. The set abstraction layer can be divided into a sampling, grouping, and PointNet layer. The sampling layer adopts the farthest point sampling (FPS) point sampling method: first, an initial point is selected and the distance between the initial point and other points is calculated. The point with the largest distance is selected to be added to the down sampling point set, and then the distance is updated. Then, the cyclic update is iterated down until the target number of sampling points is obtained. It can be understood as selecting a center point from N points. This will converge faster than random sampling"Its internal structure is shown in Fig. 2.

3.1.1. Sampling layer

The sampling layer mainly samples the original input point cloud data, selects the center point of the local area, and samples the input $N\times(d+c)$ point matrix to obtain an $n\times(d+c)$ sampling point matrix. Where N is the number of input points, n is the number of output points, d is the coordinate dimension, and c is the feature dimension. To sample the surface of the point cloud to generate a set of points with a relatively uniform distribution so that the sampling result has a more obvious contour in space, PointNet++ uses the farthest point sampling algorithm for sampling instead of random sampling.

3.1.2. Grouping layer

The grouping layer mainly combines the local regions of the sampling points. For the input $n\times(d+c)$ sampling point matrix, each sampling point is taken as the center of the circle. A local region is constructed according to the circle's center and a specific region's radius. The local feature points are sampled in each region to obtain the point set $n\times k\times(d+c)$ matrix divided into local regions. Among them, n is the number of center points and the number of local area point sets groups, while k is the number of points in each group. When the feature points in the local scope are insufficient, the aggregation is completed by repeatedly selecting the feature points. The above steps are repeated to obtain the sampling clusters under different radii, and then matrix splicing is performed after maximum pooling of the sampling clusters under the radii.

3.1.3. PointNet layer

The PointNet layer mainly uses PointNet to extract the local features of the local areas under different radii output by the grouping layer to obtain the corresponding feature vectors so that the following fully connected layer can process the classification task.

new_xyz: The coordinates of 512 center points obtained after sampling

Idx: is the index of points within each region

grouped_xyz: The grouped point set is a four-dimensional vector (*batch_size*, 512 regions, 32 points in each region, and three coordinates for each point)

3.2. Fully connected layer

The fully connected layer of PointNet++ follows the fully connected layer structure of PointNet, and its structure diagram is shown in the following figure. The fully connected layers of PointNet++ are the linear layer, normalized layer, regularization layer, linear layer, normalized layer, regularization layer, and linear layer.

3.2.1. Linear layer

It is mainly used for size transformation of the matrix, that is, to convert the input matrix into an appropriate size matrix to facilitate the subsequent processing of the neural layer as follows:

$$y = xA^T + b \tag{1}$$

where x is the input matrix, A is the weight matrix, and b is the offset.

3.2.2. Normalized layer

Its main function is to constrain the scale range of the input characteristic data to maintain a good distribution and ensure the efficiency of model training. When the normalization layer is not used, the subtle changes in the shallow parameters will be amplified after multiple layers of linear changes and activation functions, thus changing the input distribution of each layer and causing the deep network to adjust and adapt continuously. This easily causes the training parameters to fall into the saturation region, making it difficult for the model to converge. This phenomenon is called ICS (internal covariate shift). The BN layer proposed in is the normalization layer, which avoids the case where the training parameters fall into the saturation region from the perspective of changing the data distribution.

If the input batch data are x: $B = \{x_1, x_2, \ldots, x_m\}$, first calculate the mean and variance of the batch data through the following two formulas.

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2}$$

$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i-\mu_B)^2\,, \tag{3}$$

where x_i are the batch data, m is the total number of data batches, μ_B is the average value of the batch data, and σ_B^2 is the batch data variance.

After the mean and variance of the batch data are obtained, the mean and variance removal operation (normalization) is performed as follows:

$$\widehat{x_i} \leftarrow \frac{x_i-\mu_B}{\sqrt{\sigma_B^2+\in}} \tag{4}$$

where x_i is the batch data, μ_B is the average value of the batch data obtained by formula (2), σ_B^2 is the variance of the batch data obtained by formula (3), and $\in$ is a small positive number used to avoid zero value of the batch data variance.

Finally, the scale transformation and offset are carried out by the following formulas to prevent the data from being restricted to a normal distribution due to the normalization process, which will lead to a decline in the expressive ability of the neural network and restore the expressive ability of the data.

The BN layer limits the problem of magnifying small parameter changes as the network deepens and is more adaptable to parameter changes, allowing us to use a larger learning rate to accelerate the convergence of the model. At the same time, using the BN layer also avoids the problem of overfitting to a large extent.

3.2.3. Regularization layer

Additionally, within a layer referred to the dropout layer, some neurons are inactivated through a set probability to make them unable to function. By randomly deactivating neurons, the complexity of the neural network can be simplified, the dependence of the deep network on the shallow network can be reduced, and overfitting can be alleviated. The calculation formulas of the neural network before adding the dropout layer are as follows:

$$z_i^{(l+1)} = w_i^{(l+1)}y_i^l + b_i^{(l+1)} \tag{5}$$

$$y_i^{(l+1)} = f\left(z_i^{(l+1)}\right) \tag{6}$$

In formula (5), y_i^l is the output of layer l, $w_i^{(l+1)}$ is the weight parameter of layer $l+1$, and $b_i^{(l+1)}$ is the offset of layer $l+1$.

In formula (6), $z_i^{(l+1)}$ is the linear change result of layer l obtained by formula (5), f is the activation function, and $y_i^{(l+1)}$ is the output of layer $l+1$.

The calculation formulas of the neural network after adding dropout are as follows:

$$r_j^{(l)} \sim Bernoulli\,(p) \tag{7}$$

$$\tilde{y}^{(l)} = r^{(l)} * y^{(l)} \tag{8}$$

In formula (7), p is the neuron inactivation probability, *Bernoulli* is the Bernoulli function, and $r_j^{(l)}$ is a random vector (equal to 0 or 1), which is used to remove neurons randomly.

In formula (8), $r^{(l)}$ is the random vector obtained by formula (7), $y^{(l)}$ is the output of layer l, and $\tilde{y}^{(l)}$ is the dropout result of the layer l output.

3.3. Class scores

The class scores layer is the final layer in the PointNet++ neural network, composed of multiple softmax layers. The role of the softmax layer is to map the score of the input point cloud to a probability distribution. For each class category, there is a corresponding weight vector. These weight vectors are then normalized through the softmax function, resulting in a distribution vector representing the probability of the corresponding class.

For example, suppose there are six classes in the point cloud {*point_1*, *point_2*, …, *point_6*}, and each class has a corresponding weight vector [$w_{\_1}$, $w_{\_2}$, …, $w_{\_6}$]. These weight vectors can be normalized through the softmax function, resulting in [$s_{\_1}$, $s_{\_2}$,…, $s_{\_6}$], where *s_i* is the probability score of *point_i* belonging to the corresponding class.

The output of the class score layer is an $N\times C$ matrix, where N is the number of points in the point cloud, and C is the number of class categories. In this matrix, each element represents the probability score of the corresponding class, and the higher the score, the higher the class probability. For example, for a point cloud containing six classes, the output of the class score layer is a 6×6 matrix, where each element represents the probability score of the corresponding class.

Through the output of the class score layer, the PointNet++ neural network can calculate the probability of each category in the point cloud to realize the point cloud classification task.

3.4. FPS algorithm improvement

In the PointNet++ network, for the set of sampling points obtained by using the farthest point sampling algorithm, take each point as the central point to sample different numbers of clusters under different radii to generate sampling clusters, splice the cluster information (including feature information and location information) under different radii, and then carry out subsequent processing. Among them, the purpose of cluster sampling with different numbers under different radii is to extract regional features. If the number of points under the sampling radius fails to meet the sampling number requirements, cluster sampling is completed by repeatedly sampling points within the radius. From the perspective of subsequent cluster sampling, the sampled cluster should represent the corresponding region as much as possible and express the information of the corresponding region as accurately as possible. When sampling clusters, we should avoid repeated sampling of points within the sampling radius to improve the effectiveness of local feature extraction of sampled clusters. For the center point of cluster sampling, there should be as many points as possible within the sampling radius.

Although the FPS algorithm can relatively evenly select the point set that replaces the whole point cloud model, it only pays attention to the distance information between points in the process of FPS, takes the distance information as the only standard for selection, and ignores the number of points around the selected point. In this way, it is inevitable to select the farthest point, and the number of points near the farthest point is small in the process of point selection. The denser the points near the sampling point, the richer the local area information, and the greater the local weight of the point. Under the condition that the sampling quantity of the farthest point remains unchanged, to avoid repeated sampling of points within the sampling radius as much as possible in the subsequent cluster sampling, we should add the selection standard of the number of points around the sampling point when sampling the farthest point so that we can select the points with the richest regional information under the condition of uniform point selection. In this regard, we focus on the richness of regional information, and based on the conventional FPS algorithm, we propose its improved version — the DR-FPS algorithm. Compared with the conventional FPS algorithm, the selection criteria of the number of points within a specified range are added to the selection of sampling points so that the improved algorithm can consider the local area information in the sampling process. Among the sampling points

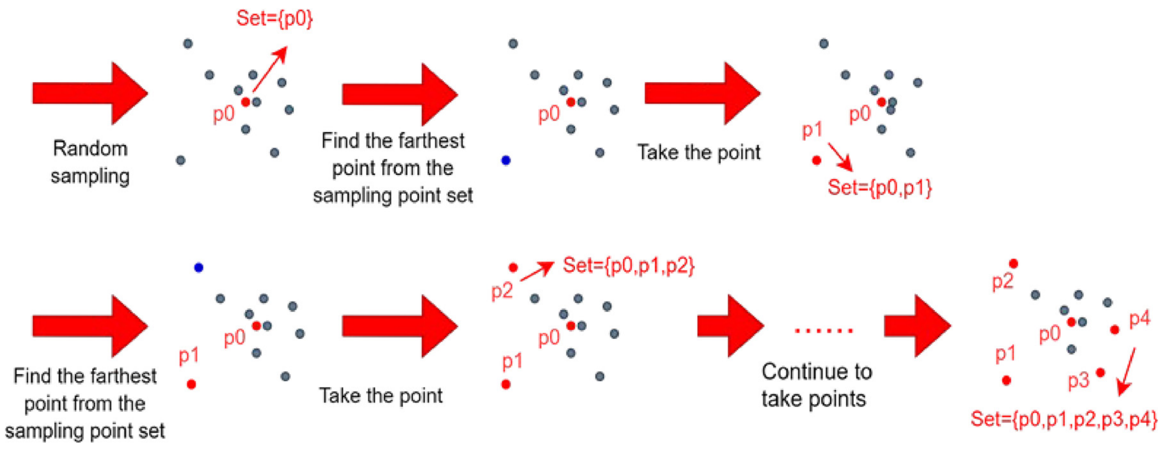


Fig. 3. Illustration of the conventional FPS algorithm.

Table 2
DR-FPS algorithm steps.

```
Algorithm 2: DR-FPS
Input: point cloud data xyz; number of sampling points n_point; density radius R
Output: sampling point set S
1   dist = distance(xyz, R)// Calculate the distance between each point
2   index = random(0, xyz.length)// Randomly select the initial sampling point $P_0$
3   S.append(xyz[index])// Add sampling points to point set S
4   L = dist [index]// Distance matrix L between all points and point set S
5   For i in range(n_point)//Take the point subscript of the first 5 distances
6       farthestIndex [5] = maxValueIndex(L, 5)//The number of points within the density radius from the top 5 points
7       count [5] = count_lessValue(dist[farthestIndex], R)//Select the highest density as the new sampling point
8       index = farthest Index[maxValueIndex(count)]// Add sampling points to point set S
9       S.append(xyz[index])
10  return S
```

with little difference in the longest distance, the point with the most abundant regional information is selected as the sampling point. The main implementation steps of the FPS algorithm are given in Fig. 3 and Table 1, while those of DR-FPS one is given in Fig. 4 and Table 2. Compared with the conventional FPS algorithm, the selected result point set of the proposed DR-FPS one has more abundant contour information. Still, the selected points are not as uniform as those of the conventional FPS algorithm. In terms of the time cost of the algorithm, the DR-FPS algorithm calculates the Euclidean distance between all points of the point cloud data and performs additional sorting when selecting the sampling points. In contrast, the conventional FPS algorithm only needs to calculate the Euclidean distance from some points to all points. Thus, the DR-FPS algorithm has more time overhead than the conventional FPS one.

Because in the process of experimentation, we selected 3 points, 5 points, 7 points and 9 points for experiments, and found that 3 points are not enough to obtain all the features of the farthest point, so the loss of the segmented image is large. If 7 and 9 points are selected, the accurate image information will not be

The first layer is set abstraction layer. Assuming the input point cloud data is (16, 1024, 3), which means a sample of 1024 points with only *xyz* coordinates. Send it to the first layer set abstraction layers. Parameters set

xyz: Point containing only coordinates;

points: Not only includes coordinates, but also features extracted from each point after passing through the previous layer;

n_point = 512: Sample layer finds 512 points as the center point, which is a manual and experiential selection;

Radius = 0.2: The radius of the ball square in the Grouping layer is 0.2. Note that this is the scale after coordinate normalization;

n_Sample = 5: Samples are taken around each center point within a specified radius sphere, with an upper limit of 5

Mlp = [64, 64, 128]: PointNet layer has 3 layers, with feature dimension changes of 64, 64, 128

new_xyz: the coordinates of 512 center points obtained after sampling

farthestIndex is the index of points within each region

S: the grouped point set is a four-dimensional vector (batch_size, 512 regions, 5 points in each region, and 3 coordinates for each point)

Abstract layer is the key part of PointNet++ neural network. It extracts features by sampling and abstracting the point cloud. It has the following shortcomings: The traditional FPS algorithm, which realizes the uniform sampling of the point cloud model to obtain the sampling point set, so as to extract the overall features of the point cloud, but when there is interference data in the point cloud, the algorithm cannot be well removed for sampling, this paper improved on the traditional FPS algorithm.

In the complete connection layer, the feature vector output by the set abstraction layer is fully connected to generate the final output label. It has disadvantages that may lead to overfitting. Complete connection layer needs to connect all feature vectors, so overfitting problem may occur in the training process. If the network depth is too deep, it may lead to more serious overfitting problems.

Class scores is the final output tag that is softmax manipulated to produce a category score. It has no confidence evaluation ability and cannot give the confidence of the predicted results.

Compared with the traditional Farthest Point Sampling algorithm, the DR-FPS algorithm in this paper considers the point density characteristics of each point when carrying out feature sampling on the point cloud data. With the same amount of data, the point density dependent apothecary sampling algorithm has a stronger ability to extract the overall feature of the point cloud model. The traditional FPS algorithm can evenly sample the point cloud model, and the sampling point set is evenly distributed in the origin cloud model. However, the traditional

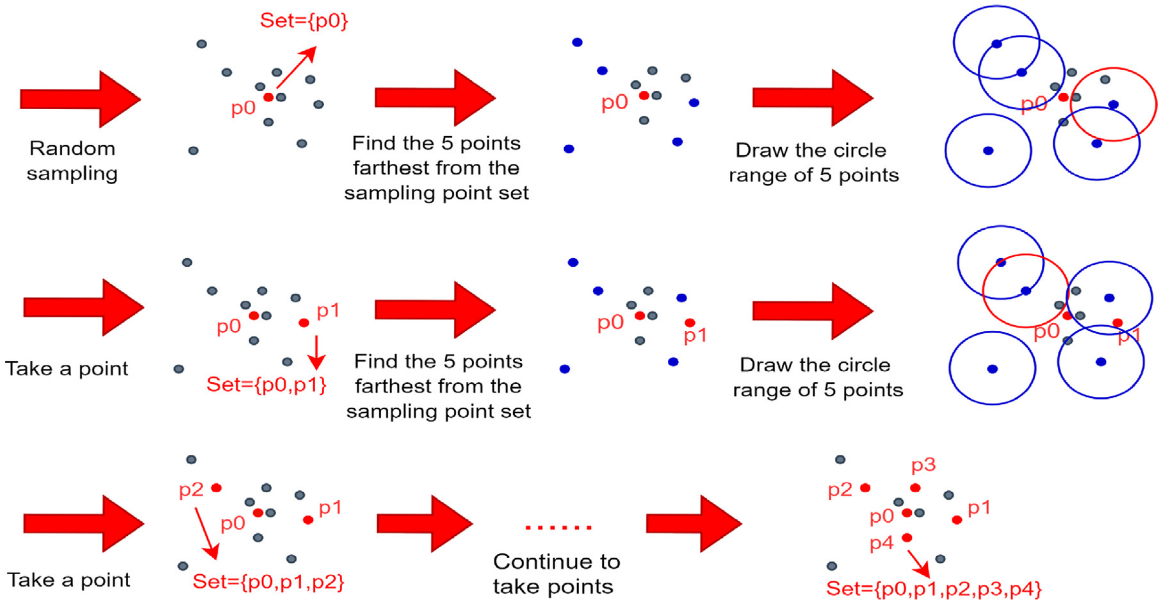


Fig. 4. Illustration of the proposed DR-FPS algorithm.

FPS algorithm only focuses on the distance relationship between points, thus ignoring the representation of point density in the point cloud characteristics. Based on this, we propose the point DR-FPS algorithm. Through experimental verification, we found that it has better ability to extract and recognize point cloud data with complex point information.

4. Experimental

4.1. Experimental dataset

This study adopted the ModelNet10 and ModelNet40 datasets as the training and test data of the 3D point cloud experiment. The data comprised 3D point cloud models drawn by CAD, including some common items from daily routine. The 3D point cloud data structure was clean and tidy, which is the current mainstream 3D point cloud neural network dataset. The experiments mainly covered 5-category and 10-category datasets. The former's ModelNet40 dataset contained five types of objects (desk, bathtub, bed, bench, and laptop), while the latter's ModelNet10 dataset had ten categories of items (bathtub, bed, chair, desk, dresser, monitor, nightstand, sofa, table, and toilet). To facilitate training and testing of the actual progress of the improved model, we simplified the original dataset in the number of point cloud instances. For the 5- and 10-category datasets, we selected 100 items of point cloud data for each type of object and then divided them into training and test sets in a 7:3 ratio for training. In addition, we also prepared the corresponding classification test set to test the model's accuracy after training specifically. The number of point cloud data points of 10 classification objects was 50, 100, 100, 86, 86, 100, 86, 100, 100, and 100, and the number of point cloud data points of 5 classification objects was 216, 86, 445, 103, and 99.

We pre-processed the raw data of the 3D point cloud according to the common methods in the PointNet++ neural network. Before training, each point cloud instance of the dataset was randomly scaled, moved, and removed to avoid its great impact on the dataset. The amplitude of the preprocessing operation was controlled in a small range, and then the pre-processed point cloud data were incorporated into the neural network classification model. The training model had improved robustness and generalization ability due to the dataset pre-processing.

4.2. Model parameters and experimental environment

The Adam optimizer was used to dynamically adjust the parameters during training, where the initial learning rate was set to 0.001, and the weight decay was set to 0.0001. The optimizer learning rate setting was dynamically adjusted, the attenuation step size was set at 20, the attenuation coefficient was set at 0.7, and the learning rate was dynamically adjusted to 70% of the initial learning rate every 20 rounds of training. In the DR-FPS algorithm, the density calculation radius was set to 6.0.

All training and testing in this paper were carried out in the Windows 10 environment (64-bit system), and the hardware used an NVIDIA GeForce GTX 1650 graphics card GPU with 4 GB of video memory. Open-source libraries such as PyTorch and NumPy were used to build neural networks. The coding environment was Python 3.9, and Pycharm 2022.1.2 was used for programming.

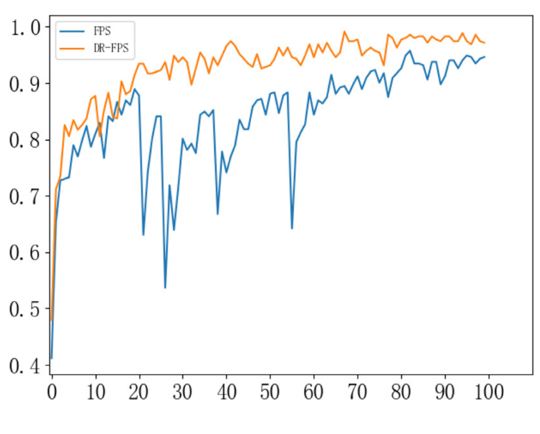
4.3. Analysis of results

We trained PointNet++ based on the conventional FPS and proposed DR-FPS algorithms 5- and 10-category datasets, respectively. The training results are shown in Fig. 5.

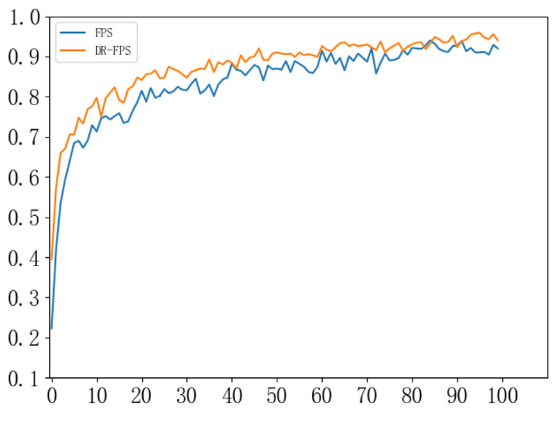
In terms of accuracy, the DR-FPS algorithm was more accurate in the initial stage of training than the conventional FPS one. A higher curve in Fig. 5 indicated a more powerful learning ability of the DR-FPS algorithm. In general, its accuracy also exceeded that of FPS, indicating that under the same training rounds, the PointNet++ neural network with the DR-FPS algorithm obtained more feature information than the conventional FPS. After 100 rounds of training, the accuracies of the conventional FPS and

DR-FPC algorithms for five categories were 0.946023 and 0.989529, respectively, differing by 4.35%. For ten categories, these values were 0.920000 and 0.972000, differing by 5.2%.

In terms of the loss value, the loss value of the DR-FPS algorithm in the training process was generally lower than that of the conventional GPS one. As shown in Fig. 6, the former's fluctuation was less, the overall was more stable, and the curve dropped faster. After 100 rounds of training, the loss values of the FPS and DR-FPS algorithms for 5-category classification were 0.157804 and 0.071051, while those for ten-category classification were 0.248888 and 0.206696, respectively. Thus, the improved algorithm reduced the training loss values of 5- and 10-category

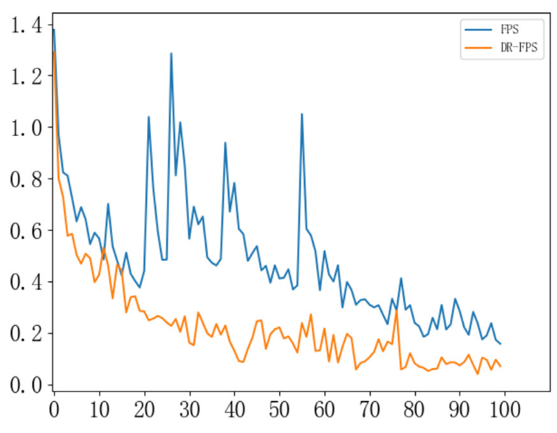


(a)

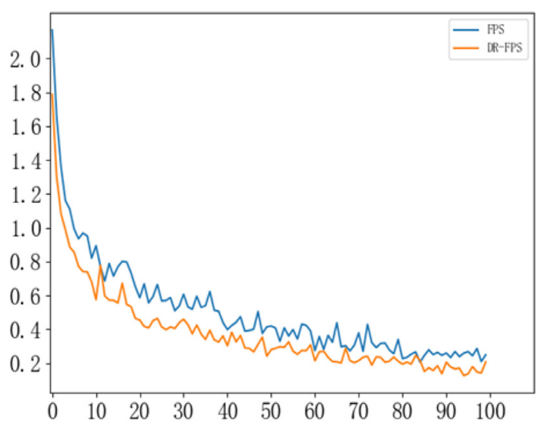


(b)

Fig. 5. Model classification accuracy: (a) 5 categories; (b) 10 categories.



(a)



(b)

Fig. 6. Model training loss values: (a) 5 classifications; (b) 10 classifications.

Table 3
The average accuracy of the model test (5 categories).

Sampling algorithm	Instance accuracy	Class accuracy
FPS	0.889	0.879
DR-FPS	0.915	0.913

Table 4
The average accuracy of the model test (10 categories).

Sampling algorithm	Instance accuracy	Class accuracy
FPS	0.949	0.948
DR-FPS	0.953	0.970

classifications by 0.086753 and 0.042192, respectively, compared to FPS.

After training, the model's accuracy was tested several times using the 5- and 10-category test sets, which were larger than the training dataset. The averaged test results are listed in Tables 3 and 4

The DR-FPS algorithm outperformed the conventional FPS one by 4% in average instance accuracy and by 2% in average class accuracy for 5-category tasks. These improvements were 2.6 and 2.4%, respectively, for 10-category tasks. The improvement could be more pronounced if more classification tasks were trained.

The classification accuracies of the DR-FPS and FPS algorithms for each particular type of objects under different classification are shown in Tables 5 and 6. As seen from the above tables, in most categories, the average classification accuracy of the DR-FPS algorithm exceeded that of the conventional FPS algorithm.

The time consumption of FPS and DR-FPS algorithms in various classifications is listed in Table 7. It can be seen that with

Table 5
The average accuracy for each type of object test (5 categories).

Category	FPS accuracy	DR-FPS accuracy
Desk	0.990	1.000
Bathtub	0.977	1.000
Bed	0.909	0.917
Bench	0.962	0.942
Laptop	0.901	1.000

Table 6
The average accuracy for each type of object test (10 categories).

Category	FPS accuracy	DR-FPS accuracy
Bathtub	0.951	0.980
Bed	0.951	0.980
Chair	1.000	1.000
Desk	0.886	0.922
Dresser	0.818	0.791
Monitor	0.970	0.998
Nightstand	0.806	0.855
Sofa	0.980	0.996
Table	0.500	0.658
Toilet	0.990	0.980

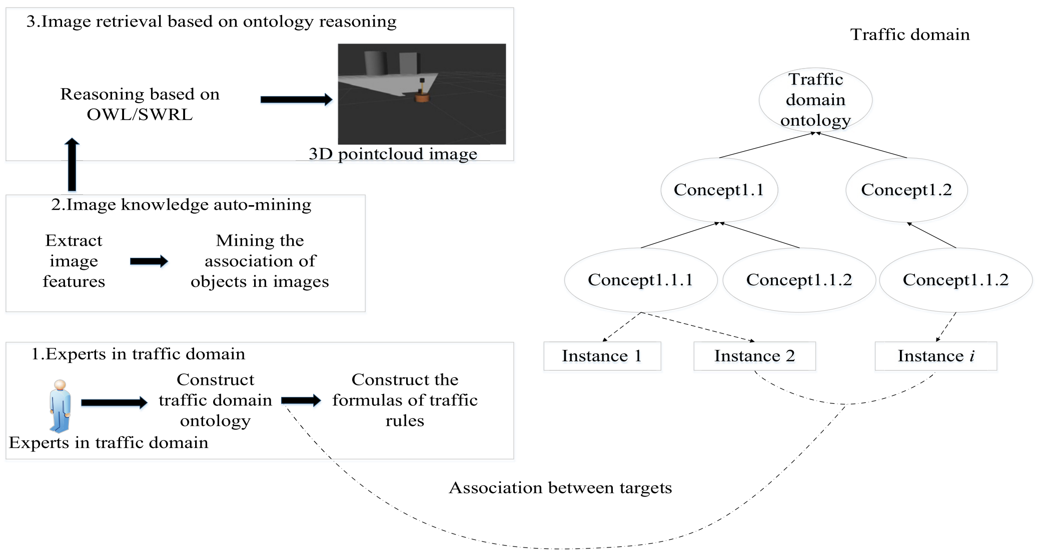

3.Image retrieval based on ontology reasoning
Reasoning based on OWL/SWRL
3D pointcloud image
2.Image knowledge auto-mining
Extract image features
Mining the association of objects in images
1.Experts in traffic domain
Construct the traffic domain ontology
Construct the formulas of traffic rules
Experts in traffic domain
Traffic domain
Traffic domain ontology
Concept1.1
Concept1.2
Concept1.1.1
Concept1.1.2
Concept1.1.2
Instance 1
Instance 2
Instance i
Association between targets

Fig. 7. Semantic retrieval framework for traffic images based on domain knowledge images.

Table 7
Time consumption of FPS and DR-FPS algorithms in various classifications.

Number of categories	FPS time consumption	DR-FPS time consumption
5	164 min	172 min
10	377 min	382 min
15	525 min	531 min

an appropriate increase in time consumption, the DR-FPS algorithm significantly improves its accuracy. The experimental results show that the proposed PointNet++ neural network based on the DR-FPS algorithm performs well in classification tasks. Compared with the conventional FPS algorithm, it shows better local feature extraction ability in the classification task and further strengthens the relationship between the global feature and local feature of the 3D point cloud.

The application scenarios of our algorithm are very wide. For example, in the field of unmanned driving, the point cloud data scanned by lidar usually contains some interference, such as the dust at the rear of the car. These impurities will affect the sampling results of the traditional FPS algorithm, while the DR-FPS algorithm can better filter these sparse points and reduce the interference brought by them (Zhao et al., 2022).

This paper first used a machine learning method to identify specific targets, labeled them as instances in the traffic domain ontology, and then analyzed the spatial location relationship between specific targets. Then, through semantic reasoning technology, it judged whether the automatically labeled ontology instances meet the constraints of traffic rules. Finally, it predicted whether pedestrians were crossing the road and whether the car had to slow down. The traffic image semantic retrieval method based on target self-recognition proposed in this paper included three main steps, as shown in Fig. 7.

(1) Knowledge construction in the field of road traffic. Experts in the field build road traffic logical reasoning to obtain image retrieval results.

Domain ontology, including concept, instance, inheritance, instantiation, attribute association, and building foundation.

The formula of road traffic rules based on first-order logic.

(2) Automatic mining of image targets. Automatically identifies queries from image data.

The specific target of the request and the spatial and positional binary relationship between the targets are obtained.

(3) Image retrieval based on domain ontology reasoning. Established according to step 1.

Domain knowledge and the specific target and its association are automatically mined in step 2.

The 3D point cloud instance mapping domain ontology method is illustrated in Fig. 8. After identifying the traffic target instance in the image, the matplotlib library (http://matplotlib.org/) in Python tools gives the boundary area of the target instance and then calls the man-size function to give four parameters of the active window of each target instance: the position of the quadrilateral X axis, Y axis, width W, and height *h*. Call the rectangle function through the above four parameters to obtain the active window of the target instance. At the same time, call the Neighbors function to find other target instances adjacent to the current target instance and obtain the active windows of other target instances similarly. Window position relationship judgment algorithm: Suppose that the active windows of two target instances in the image are quadrilateral R1 and R2, which correspond to ontology instances A and B in the traffic domain ontology, respectively, and that A and B meet the direction position matrix model; then, the association relationship is returned (see Tables 8–10).

According to the comparison of data in Figs. 8, 9, and 10, the accuracy of our method in Scenario 1 is 5.3% higher than that of the FPS method; Scenario 2 has a high accuracy of 6.4; Scenario 3 has a high accuracy of 6. 5%. The recall of our method in Scenario 1 is 9.3% higher than that of the FPS method; the recall of our method in Scenario 2 is 12.7% higher than that of the FPS method; the recall of our method in Scenario 3 is 6.8% higher than that of the FPS method. Based on the above data, it can be concluded that the combination of this method with intelligent driving can improve the recognition and safety of images.

The DR-FPS algorithm proposed in this paper can extract the feature information of point clouds more efficiently in some scenes with certain interference and complex point cloud information. For example, in the automatic driving scenario based on lidar scanning point cloud, the actual car point cloud data scanned by lidar is usually not so clean, and may be mixed with other point information such as dust raised by cars. If the traditional FPS algorithm is adopted, the sampled point set contains most of the dust point information. The point density will be taken into consideration when the point cloud data is sampled by the DR-FPS algorithm which is related to the point density, which can

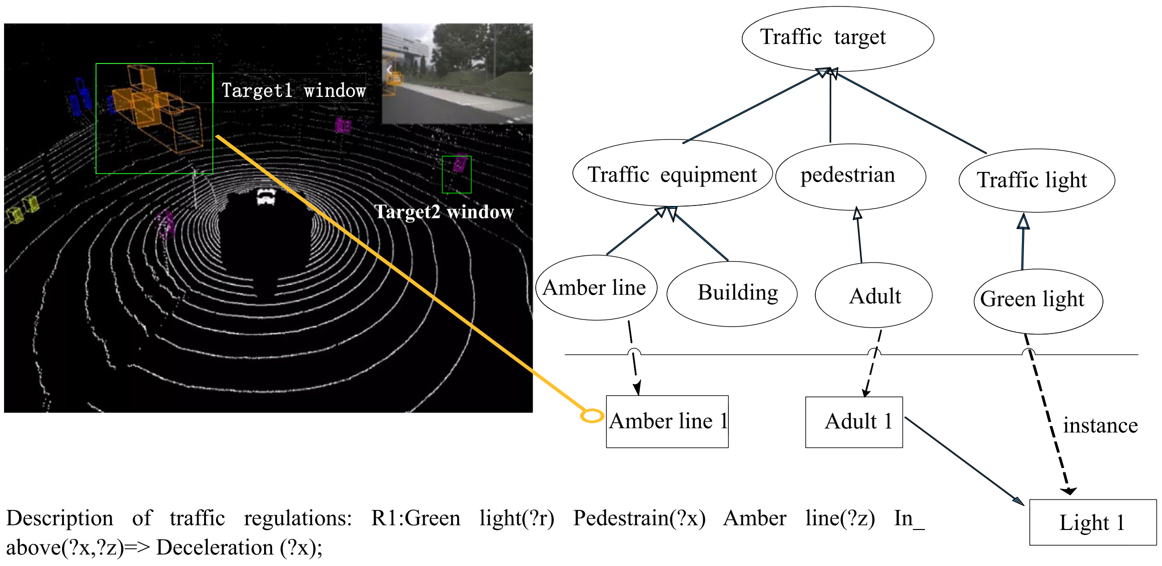


Fig. 8. 3D point cloud instance domain ontology method.

Table 8
Accuracy of FPS and DR-FPS algorithms in different scenarios.

Scene	FPS recognition accuracy	DR-FPS recognition accuracy
Simulated traffic environment 1	0.891	0.944
Simulated traffic environment 2	0.863	0.927
Simulated traffic environment 2	0.841	0.906

Table 9
Recall of FPS and DR-FPS algorithms in different scenarios.

Scene	FPS recognition recall	DR-FPS recognition recall
Simulated traffic environment 1	0.733	0.826
Simulated traffic environment 2	0.704	0.831
Simulated traffic environment 2	0.784	0.852

Table 10
F1 of FPS and DR-FPS algorithms in different scenarios.

Scene	F1 of FPS recognition	F1 of DR-FPS recognition
Simulated traffic environment 1	0.801	0.881
Simulated traffic environment 2	0.771	0.875
Simulated traffic environment 2	0.825	0.863

effectively reduce the interference caused by the point information of other objects such as soot, so as to obtain a cleaner point cloud model and enable the neural network to obtain richer point cloud characteristic information from the point cloud data of the same scale. In the training model and practical application, the point cloud features can be extracted more efficiently, and the processing efficiency and recognition ability of neural network can be improved.

By improving the FPS algorithm, this algorithm has better anti-interference function when sampling and abstracting the point cloud model, and the sampled point cloud data is relatively clean. The application scenarios of our algorithm are very wide. For example, in the field of unmanned driving, the point cloud data scanned by lidar usually contains some interference, such as the dust at the rear of the car. These impurities will affect the sampling results of the traditional FPS algorithm, while the DR-FPS algorithm can better filter these sparse points and reduce the interference brought by them. Through experimental tests, we found that the PointNet++ neural network model based on the DR-FPS algorithm had certain improvements in accuracy and recall rate.

We first discuss the threats to external validity, meaning the experimental results have universal generalization. It mainly considers two points: (1) Whether the sample is representative of all cases. (2) Whether the conclusion can be generalized to other cases. We compared the proposed and baseline methods for the first scenario on 5- and 10-category data. The results showed that the selected samples for our method are representative of all cases. We ran the DR-FPS and baseline methods for the second issue using the intelligent car platform in a simulated traffic environment.

5. Conclusions

The object recognition results of the proposed density related-farthest point sampling (DR-FPS) algorithm prove its high accuracy rate in the actual scene. Therefore, the improved method in this paper can be applied to the analysis and classification of common scenes in life. It can provide more reliable results for classification in complex environments.

This method can also be applied in the following scenarios: the artificial intelligence based on the 3D point cloud model scanned by the lidar can accurately identify the facial and oral features of the target person, and complete the nucleic acid sampling through the robotic arm, which can be a good substitute for manual sampling. Therefore, automated nucleic acid sampling has a wide range of application prospects. In the automated nucleic acid sampling, the most critical technology is pointing cloud registration, and the existing point cloud registration algorithm can achieve high and low overlap rate object image registration accuracy of 0.001 m, and the impact of overlap rate is very small. It has high security in the process of application and operation. We consider integrating Gumbel Subset Sampling in future work. it is mainly replacing FPS in Pointnet++with Gumbel Softmax to calculate the importance of each point, and then selecting down sampling points based on this probability.

CRediT authorship contribution statement

Yi Zhao: Conceptualization, Methodology, Software, Visualization. **Hao Chen:** Data curation, Writing – original draft, Investigation. **Liang Zen:** Software, Validation. **Zhao Li:** Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yi zhao reports administrative support was provided by Guangdong Ocean University. Yi zhao reports a relationship with Guangdong Ocean University that includes: board membership. Yi zhao has patent licensed to yi zhao.

Data availability

No data was used for the research described in the article.

Acknowledgment

Guangdong Intelligence Platform of Prawn Modern Seed Industry, Project number: 2022GCZX001.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jss.2023.111794.

References

Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., et al., 2015. ShapeNet: An information-rich 3D model repository. http://dx.doi.org/10.48550/arXiv.1512.03012.

Deng, Z., Yang, D., Zhang, X., Dong, Y., Liu, C., Shen, Q., 2020. Real-Time Image Stabilization Method Based on Optical Flow and Binary Point Feature Matching. Multidisciplinary Digital Publishing Institute, http://dx.doi.org/10.3390/electronics9010198.

Feng, Y., Zhang, Z., Zhao, X., Ji, R., Yue, G., 2018. GVCNN: Group-view convolutional neural networks for 3D shape recognition. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE, http://dx.doi.org/10.1109/CVPR.2018.00035.

Guo, Q., Jin, S., Li, M., et al., 2020a. Application of deep learning in ecological resource research: Theories, methods, and challenges. Sci. China Earth Sci. 63, 1457–1474. http://dx.doi.org/10.1007/s11430-019-9584-9.

Guo, Y., Wang, H., Hu, Q., Liu, H., Bennamoun, M., 2020b. Deep learning for 3d point clouds: a survey. IEEE Trans. Pattern Anal. Mach. Intell. PP (99), 1. http://dx.doi.org/10.1109/TPAMI.2020.3005434.

Hartley, R.I., Sturm, P., 1995. Triangulation. Comput. Vis. Image Underst. 68 (2), 146–157. http://dx.doi.org/10.1007/3-540-60268-2_296.

He, C., Gong, J., Yang, Y., Bi, D., Lan, J., Qie, L., 2021. Real-time track obstacle detection from 3d lidar point cloud. J. Phys. Conf. Ser. 1910 (1), 012002. http://dx.doi.org/10.1088/1742-6596/1910/1/012002, (5pp).

Li, J.L., Li, Y.T., Long, J., Zhang, Y., Gao, X.R., 2021. SAP-Net: A simple and robust 3d point cloud registration network based on local shape features. Sensors 21 (21), 7177. http://dx.doi.org/10.3390/s21217177.

Liu, Z.S., Song, W., Tian, Y.F., Ji, S.M., Sung, Y.S., Wen, L., Zhang, T., Song, L.L., Gozho, A., 2020. VB-Net: Voxel-based broad learning network for 3D object classification. Appl. Sci. 10 (19), 6735. http://dx.doi.org/10.3390/app10196735.

Mahendru, M., Kumar, D.S., 2022. Portable learning approach towards capturing social intimidating activities using big data and deep learning technologies. Int. J. Perform. Eng. 18 (2022), 668.

Maturana, D., Scherer, S., 2015. VoxNet: A 3D convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, IEEE, http://dx.doi.org/10.1109/IROS.2015.7353481.

Mavroidis, C., Flanz, J., Dubowsky, S., Drouet, P., Goitein, M., 1998. High performance medical robot requirements and accuracy analysis. Robot. Comput.-Integr. Manuf. 14 (5–6), 329–338. http://dx.doi.org/10.1016/S0736-5845(98)00022-2.

Qi, C.R., Li, Y., Hao, S., Guibas, L.J., 2017a. Pointnet++: deep hierarchical feature learning on point sets in a metric space. http://dx.doi.org/10.48550/arXiv.1706.02413.

Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2017b. Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation. IEEE, http://dx.doi.org/10.48550/arXiv.1612.00593.

Rieg, L., Wichmann, V., Rutzinger, M., Sailer, R., Geist, T., Stoetter, J., 2014. Data infrastructure for multitemporal airborne lidar point cloud analysis – examples from physical geography in high mountain environments. Comput. Environ. Urban Syst. 45 (may), 137–146. http://dx.doi.org/10.1016/j.compenvurbsys.2013.11.004.

Shi, Z.G., Zhu, M.M., Guo, B., Zhao, M.G., 2017. A photographic negative imaging inspired method for low illumination night-time image enhancement. Multimed. Tools Appl. 76, 15027–15048. http://dx.doi.org/10.1007/s11042-017-4453-z.

Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E., 2015. Multi-View Convolutional Neural Networks for 3D Shape Recognition. IEEE, http://dx.doi.org/10.1109/ICCV.2015.114.

Tanaka, K., Waki, H., Ido, Y., Akita, S., Yoshida, Y., Yoshida, T., et al., 1988. Protein and polymer analyses up to m/z 100 000 by laser ionization time-of-flight mass spectrometry. Rapid Commun. Mass Spectrom. 2 (8), 151–153. http://dx.doi.org/10.1002/rcm.1290020802.

Tang, Z., Ma, G., Lu, J., Wang, Z., Fu, B., Wang, Y., 2020. Sonar image mosaic based on a new feature matching method. IET Image Process. 14, http://dx.doi.org/10.1049/iet-ipr.2019.0695.

Tao, W., Liu, Z., Li, Y., 2010. Research and realization of the Chinese massage robot based on three-dimensional model. In: 2010 8th World Congress on Intelligent Control and Automation. IEEE, http://dx.doi.org/10.1109/WCICA.2010.5554224.

Wang, W., Zhou, H., Yan, Y., Cheng, X., Yang, P., Gan, L., Kuang, S., 2023. An automatic extraction method on medical feature points based on PointNet++ for robot-assisted knee arthroplasty. Int. J. Med. Robot. http://dx.doi.org/10.1002/rcs.2464.

Wei, H.Y., Feng, Q., Fei, H., Wei, H., 2023. Hierarchical 2D/3D alignment method based on enhanced DRR and gradient direction weighted histogram. Int. J. Perform. Eng. 19 (1), 1–9.

Zhang, J., Lin, X., 2016. Advances in fusion of optical imagery and LiDAR point cloud applied to photogrammetry and remote sensing. Int. J. Imag. Data Fusion 8 (1), 1–31.

Zhao, Y., Huang, H.C., Li, Z.X., Huang, Y.W., 2022. Intelligent garbage classification system based on improve MobileNetV3-Large. Connect. Sci. 34 (2022), 1299–1321.

Yi Zhao is a lecturer of software engineering at Guangdong Ocean University, China. He received his Ph.D. from Wuhan University in 2019. His research interests are Image Processing, Software Testing, Software Engineering.

Liang Zeng is a software engineering student from Guangdong Ocean University. His research interests are in Computer vision, image processing, data mining.

Chen Hao is a master's student at Jinan University in China. His research interests are in computer vision and image processing.

Zhao Li is a professor of software engineering at Guangdong Ocean University, China. He received his Ph.D. from Wuhan University in 2015. His research interests are Agile Software Development, Software Testing, Software Engineering.