GPFEnet: a lightweight grid parallel feature extraction net for 3D segmentation in agriculture

Anonymous Authors

***Abstract*—The real-time and accurate 3D segmentation of fruits in agricultural field is crucial for precision agriculture. In complex field scenarios, point clouds generated by depth cameras often have gaps and point dispersion problems, using the traditional farthest point sampling algorithm (FPS) makes it difficult to uniformly capture key features, and FPS requires extensive sorting operations, which reduces the speed of feature extraction. To optimize the sampling uniformity of point clouds with multiple holes in agricultural field scenes while enhancing feature extraction efficiency, this paper proposes a lightweight and efficient density-based grid parallel feature extraction network GPFEnet. First, an adaptive grid parallel grouping algorithm was designed to address the issue of low efficiency in capturing point cloud features caused by irregular point cloud data in complex field scenarios. Based on this fundamental algorithm, a density-based grid random sampling algorithm GDSample was proposed, which optimize sampling uniformity while improving sampling efficiency through support for parallel computation. Finally, the grid parallel feature extraction subnetwork GPFEnet was developed which can be embedded into point cloud deep learning networks. Testing results on field datasets show that the proposed GDSample algorithm is approximately 35 times faster than FPS, significantly accelerating feature extraction while ensuring more even coverage of key features. To confirm the superiority of GPFEnet on multi-hole fruit datasets, the GPFEnet was integrated into various cutting-edge point cloud deep learning networks, achieved a significant enhancement in speed of feature extraction while maintaining similar accuracy in fruit segmentation. The code will be publicly available after the paper is accepted.**

***Index Terms*—Point Cloud, Deep Learning, Fruit Segmentation, Lightweight Grid Parallel Feature Extraction, Parallel Computing.**

I. Introduction

With the advancement of agricultural automation, how to further enhance the real-time performance of automatic fruit picking while accurately identifying fruit phenotypes and their environmental characteristics has become a core issue in current automated fruit harvesting. Currently, images and point clouds, as two primary data forms, are widely used in the agricultural field. Compared to images, point clouds can more comprehensively reflect crop phenotypes and environmental characteristics, making them an emerging research direction. However, in current agricultural point cloud research, the inefficiency of point cloud generation and recognition remains a major factor limiting the application of point clouds in agricultural automation.

In recent years, the efficiency of real-time point cloud capture and generation has been partially addressed. He et al. [1] and Yang et al. [2] achieved significant improvements across multiple efficiency benchmarks. In complex field scenarios, image reconstruction based on depth cameras has overcome many shortcomings of traditional 3D scanners, becoming the mainstream method for acquiring crop point clouds. Depth cameras are inexpensive, highly portable, and capable of effectively capturing plant features, ensuring the completeness of plant trait reconstruction, and hold great potential for large-scale applications in agriculture. However, due to environmental influences, the fruit point clouds obtained by this method often contain many discrete points with complex backgrounds (such as leaves, branches, and lighting), as well as missing points and holes. Therefore, how to accurately identify point clouds in the presence of multiple discrete points, holes, and missing data remains an urgent problem to be solved.

With the development and application of deep learning in point clouds, the accuracy of point cloud recognition has gradually improved. On public datasets, model recognition accuracy has made significant progress. Taking the Stanford University Indoor 3D Dataset (S3DIS) as an example, after the emergence of Point Transformer [3], the model's mean Intersection over Union surpassed 70%. In field scenarios, Li et al. [4] developed DeepSeg3DMaize based on PointNet++, achieving a segmentation accuracy of up to 91% for corn stems and leaves. Liu et al. [5] proposed a density-based feature extraction and feature propagation method, achieving a segmentation accuracy of up to 95% on apples. Therefore, under the current research background, deep learning models for point clouds are capable of handling recognition tasks in most scenarios. However, the cost behind this high-precision recognition is a sharp increase in computational load. For instance, a simple PointNet++ [6] end-to-end run requires about 2.0 GFlops, but on Stratified-Transformer [7], the computational load reaches nearly 30.0 GFlops. The lengthy computation time results in low point cloud recognition efficiency on low-power field devices, leading to poor immediacy and human-machine interaction in agricultural automatic picking.

The low efficiency of point cloud recognition is primarily attributed to two factors: the excessive number of neurons and the frequent invocation of the point cloud feature extraction process, which is difficult to reduce while maintaining recognition accuracy. Therefore, optimizing the operational efficiency of feature extraction has become an effective approach to enhance the efficiency of point cloud recognition. Most existing classical models [3], [6], [7], [8], [9] employ the FPS algorithm to iteratively select the point farthest from the currently selected set as the key feature. Although FPS is relatively easy to implement, it yields lower uniformity of key features for datasets with discrete points and multiple

holes, such as fruit datasets. When reducing the number of sampling points, FPS tends to concentrate the sampling points on the edges of the point cloud, which significantly lowers the recognition accuracy in cases of multiple discrete points and holes. Additionally, FPS determines the farthest point through extensive sorting, resulting in a complexity of $O(N^2)$, which inevitably reduces the efficiency of feature extraction.

In order to address the issues of low sampling uniformity and inefficient feature extraction in complex agricultural scene datasets using traditional sample methods, thereby improving the real-time performance of point cloud deep learning in agricultural harvesting, A Grid and Density-based Parallel Feature Extraction Network (GPFEnet) was proposed. The main contributions are as follows:

- A grid-based parallel grouping algorithm with point cloud shape adaptability was proposed, which eliminate the need for distance calculations between points, thereby enhancing the efficiency of the grouping algorithm.
- A density-based grid random sampling algorithm, GDSample, was proposed, which addresses the issues of low sampling efficiency and difficulty in uniformly covering key features in complex field scenarios with multiple holes and discrete points.
- A grid parallel feature extraction framework, GPFEnet, was designed and embedded into classic point cloud neural networks. This framework improves the feature extraction efficiency of traditional networks for point clouds of porous fruits while maintaining crop segmentation accuracy.

II. Related Work

Optimizing the uniformity of point cloud sampling and the efficiency of feature extraction is of significant importance for the real-time recognition of fruits in complex field environments, and directly impacts the accuracy of segmentation. This paper will describe research progress in two aspects: deep learning in fruit point cloud segmentation and point cloud feature extraction techniques.

A. Deep Learning in Fruit Point Cloud Segmentation

In recent years, the rapid advancement of 3D point clouds in computer vision, coupled with improved spatial learning capabilities in point cloud semantic segmentation for industrial applications, has prompted researchers to explore its use in agricultural settings for fruit recognition and segmentation.

Li et al. [4] developed the DeepSeg3DMaize system based on PointNet, offering a framework for automated analysis of 3D phenotypic features at the individual plant level. Chen et al. [10] enhanced RandLA-Net's local feature aggregation module, enabling semantic segmentation of large-scale agricultural scenes using 3D point clouds. Jayakumari et al. [11] improved PointNet's random sampling scheme, facilitating the segmentation of cabbage and tomato. Hu et al. [12] applied an HSV color enhancement algorithm to PointNet++ for segmentation of rapeseed point clouds. Yu et al. [13] introduced a 3DSTN spatial transformation network to align point clouds on PointNet and designed a pyramid pooling module for feature extraction, achieving successful segmentation of apples, pears, and lemons. Liu et al. [5] developed FSDnet, a network that leverages Gaussian density feature extraction and feature propagation, improving the semantic segmentation accuracy of strawberry and apple point clouds.

As point cloud-based fruit segmentation accuracy improves in field tasks, deep learning models have become increasingly complex, reducing operational efficiency. Moreover, the irregular shapes of point clouds from depth cameras, their high complexity, and the presence of numerous discrete points and gaps have hindered the widespread adoption of point cloud deep learning in agriculture. Thus, efficiently extracting key features from porous datasets in field conditions while preserving fruit segmentation accuracy is a critical challenge that needs urgent attention.

B. Current Status of Feature Extraction in Point Cloud Deep Learning

With the advancement of deep learning technologies for point clouds, significant progress has been made in optimizing point cloud feature extraction. Gradually, four mainstream feature extraction methods have emerged, based on projection, voxelization, deep learning, and raw point clouds.

Projection-based Feature Extraction. To accomplish object detection tasks, many studies project 3D point clouds onto 2D images. Chen et al. [14] combined features from different viewpoints to achieve accurate 3D object detection in autonomous driving. Similarly, Lang et al. [15] proposed a new encoder that converts point clouds into a format suitable for detection pipelines. However, this projection process often results in the loss of important geometric details from the point clouds.

Voxel-based Feature Extraction. Yilun et al. [16] voxelized point clouds into 3D grids and generated a small number of high-quality initial predictions through lightweight convolutional operations. Graham et al. [17] voxelized point clouds and applied 3D CNNs to process spatially sparse data more efficiently. Additionally, Le [18] designed PointGrid, which better represents local geometric details through voxel grids. However, voxelization is computationally intensive and more complex compared to point-based extraction methods.

Deep Learning-based Feature Extraction. Dovrat et al. [19] introduced the generator-based S-Net (GS) for feature extraction, while Lang et al. [20] enhanced it with a soft projection operation, resulting in the SampleNet network, which excelled in classification and reconstruction tasks. Abid [21] proposed the Continuous Relaxation-based Sampling (CRS) method, utilizing reparameterization for continuous domain sampling, but its large weight matrix incurs high memory costs. Similarly, Yang et al. [22] introduced Gumbel Subset Sampling (PGS), improving speed and efficiency over traditional GS methods. Nezhadarya et al. [23] used max-pooling for key point extraction, while Qian et al. [24] developed MOPS-Net to learn a sampling transformation matrix for

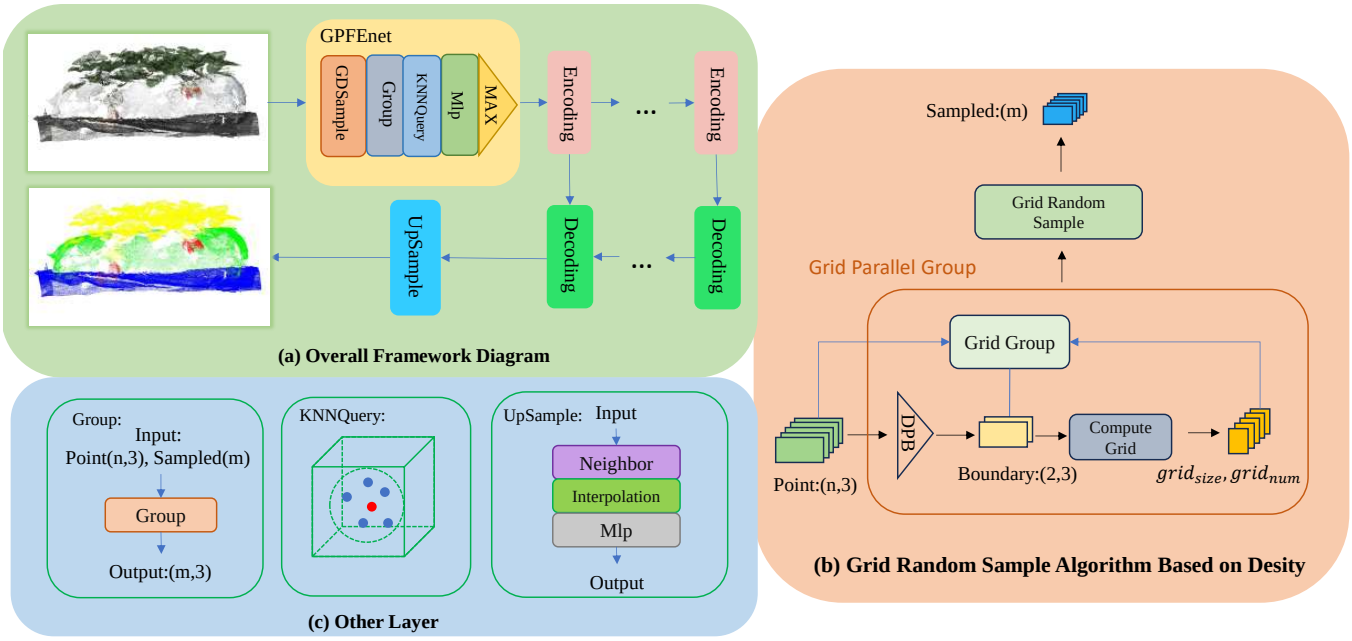

Fig. 1: The total structure of GPFEnet. (a) Overall Framework Diagram. max:max pooling. neighbor: nearest neighbor. (b) Grid Random Sample Algorithm Based on Density (GDSample). dpb: division of point cloud boundary. (c) Other Layer.

generating new point clouds. Lin et al. [25] proposed a density-adaptive method to address point cloud density variation. While these methods reduce feature loss compared to traditional algorithms, they involve high computational complexity and pre-training, making them suited only for high-precision applications.

Point-based Feature Extraction. Wei [26] introduced Poisson disk sampling (PDS) for tightly packed samples, but its use on surfaces is computationally costly due to frequent geodesic distance calculations. For uneven point cloud densities, Tokdar et al. [27] proposed the Inverse Density Importance Sampling (IDIS) algorithm, though it suffers from significant errors in density estimation in dense point clouds, limiting its applicability. To enhance efficiency, Hu et al. [12] developed RandLA-Net, a large-scale point cloud recognition network based on random sampling. Qi et al. [6] introduced PointNet++, utilizing FPS for feature extraction. Although effective for local feature aggregation and simple to implement, its dependence on multiple sorting operations hinders parallel efficiency and slows processing speeds.

With the significant enhancement of algorithm efficiency through parallel computing across various fields, scholars have introduced it into point cloud deep learning to improve model recognition efficiency. To further optimize the FPS algorithm, Zhao et al. [8] implemented CUDA parallelization of the FPS algorithm in PointWeb, significantly boosting sampling efficiency by processing point cloud blocks with multithreading. This method has been widely applied in multiple networks [3], [7], [8], [9] for feature extraction. However, this process requires sorting all unsampled points to determine the point farthest from the sampled points, which incurs substantial sorting overhead, occupying the majority of the sampling algorithm's time and preventing FPS from achieving full parallelization in feature extraction. Therefore, minimizing sorting and further parallelizing the point cloud sampling algorithm has become a critical research issue in optimizing feature extraction efficiency.

III. Methods

Traditional sampling method FPS extract key points based on the farthest distance rather than density, resulting in a higher distribution of feature points in low-density edge regions and fewer features in the central areas. This ultimately leads to an uneven coverage of key features across the entire point cloud. Additionally, FPS iteratively selects the point farthest from the currently selected set, a process that often involves extensive sorting, reducing the efficiency of feature extraction.

To address the issue of uniform key feature distribution while improving the efficiency of feature extraction, a Grid Parallel feature extraction Network (GPFEnet) was proposed, as illustrated in Figure 1. First, an adaptive grid is constructed based on the shape of the point cloud data to facilitate grid grouping, thereby avoiding distance calculations between points and enhancing grouping efficiency. Building on this, a Density-based Parallel Grid Random Sampling algorithm (GDSample) was designed. This algorithm not only extracts more features in high-density regions but also ensures that the edges of the point cloud are captured. By employing random sampling and parallel computation, it balances efficiency and accuracy, thereby improving the sampling efficiency for point clouds with multiple holes. Finally, we construct the GPFEnet

tailored for feature extraction in datasets with multiple holes and integrate it into classical point cloud deep learning models. This approach significantly enhances feature extraction efficiency while maintaining recognition accuracy for point cloud data with multiple holes.

A. Grid Parallel Group Algorithm

Traditional grouping schemes often rely on relative distance calculations and sorting operations, but the spatial distance between any two points is unknown. Consequently, in the worst-case scenario, many variables require $O(N^2)$ time to process, which directly impacts the efficiency of feature extraction from fruit point clouds.

Grid-based partitioning is an effective method to eliminate distance calculations and reduce time complexity. Ade et al. [28] utilized a value of $\mathcal{E}\sqrt{2}$ to construct a grid for 2D spatial data, reducing computation time to a linear level. Similarly, Deng et al. [29] introduced $\mathcal{E}$-neighborhood units based on grids, eliminating the need for distance calculations between points and significantly improving performance. Therefore, to enhance feature extraction efficiency, this paper constructs an adaptive grid based on the shape of point cloud data and performs parallel grid grouping, eliminating distance calculations between points and avoiding the time overhead caused by sorting, thereby achieving higher performance in grouping. This process is divided into two steps:

- **Design of Self-Adaption Grid.** Determine the boundary range of the point cloud and construct the minimum bounding box enclosing the point cloud data in 3D space. Subsequently, adaptively design the grid size, dividing the bounding box into multiple uniform, shape-adaptive cubic grids.
- **Grid Parallel Group.** Map the fruit point cloud to corresponding grid units based on the designed three-dimensional voxel grid.

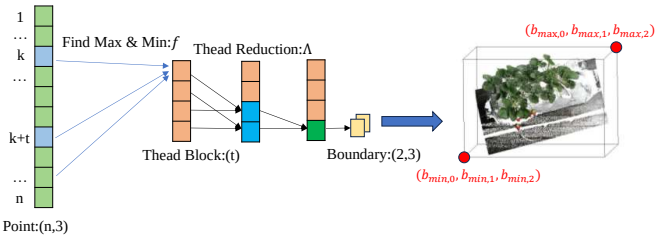

Fig. 2: Division of point cloud boundary.

1) Design of Self-Adaption Grid: In this section, we rapidly construct an adaptive grid based on the shape and sampling points of the fruit point cloud data. This is specifically achieved through the following three steps:

First, allocate threads based on the number of input point clouds. As shown in Figure 2, let the set of fruit points be: $P = \{p_{ij} \mid i = 1, 2, 3, \ldots, n;\ j = 1, 2, 3\}$. Apply for t threads as shown in Equation (1):

$$t = \min\left(1 \ll \log_2 n, 1024\right) \tag{1}$$

where $\ll$ symbol serves as a shift operator, ensuring that the number of threads is close to the size of the point cloud being processed while making the thread count a power of two, thereby enhancing the program's concurrency.

Then, obtain the minimum bounding box of the fruit point cloud. Define the overall point cloud boundary as shown in Equation (2):

$$B = \{b_{max,j}, b_{min,j} | j = 1, 2, 3\} \tag{2}$$

Obtain the boundary values of the point cloud through Equation (3):

$$b_{f,j} = \left\{ \bigwedge_{k=1}^{t+1} f\left(p_{ij}\right) \middle| \begin{array}{l} i = k, k+t, \ldots, k+\lambda t; \\ k+\lambda t \leq n; \\ k = 1, \ldots, t; \\ j = 1, 2, 3; \end{array} \right\} \tag{3}$$

where k represents the thread number, f denotes the maximum or minimum operation, and Λ means the thread reduction operation. In Equation (3), the thread numbered k processes the fruit point cloud with indices$\{k, k+t, \ldots, k+\lambda t\}$, continuing until $k + \lambda t \leq n$. Subsequently, a reduction operation is performed to obtain the boundary of the point cloud across all threads, as illustrated in Figure 2. By parallelizing the aforementioned method, the cloud boundary of the fruit point is rapidly acquired, thereby preparing the groundwork for efficient grouping.

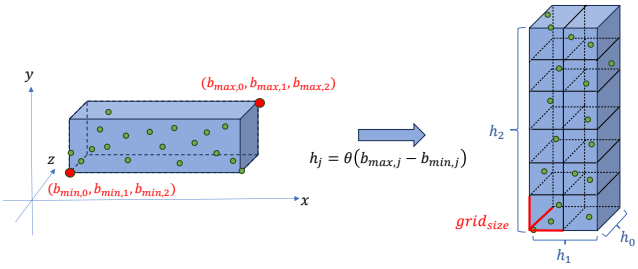

Fig. 3: Division of point cloud grid.

Finally, obtain the minimum parameters of the boundary box and the grid, including the size of the boundary, the size of grid cell and the total number of grid cells in the boundary box. As shown in Figure 3, based on the grid boundary B, the side length of the boundary box that encompasses the entire point cloud is obtained as in Equation (4):

$$h_j = \theta\left(b_{max,j} - b_{min,j}\right) \tag{4}$$

where θ is an increasing sorting function. Design the grid side length through Equation (5):

$$grid_{size} = \frac{h_1}{\sqrt[3]{\frac{m}{\prod_{j=1}^{3} \frac{h_j}{h_0}}}} \tag{5}$$

where m represents the number of points to be sampled. The total number of grid cells obtained through Equation (6):

$$grid_{num} = \prod_{i=0}^{2} \alpha \left(\frac{h_j}{grid_{size}} \right) \tag{6}$$

where α is the ceiling function. The division of the grid based on the boundary B of the fruit point cloud and the number of sampling points m, as described above, significantly ensures the robustness and flexibility of the algorithm for different fruit point cloud data.

2) Grid Parallel Group: To reduce reliance on sorting, a grid-based parallel grouping algorithm is proposed, tailored to the characteristics of fruit point cloud data and based on the constructed 3D voxel grid. This algorithm maps the point cloud to corresponding grid cells. By doing so, grouping can be performed without depending on the distance sorting of unsampled points, significantly mitigating the impact of sorting on grouping speed and effectively enhancing the operational efficiency of fruit point cloud feature extraction.

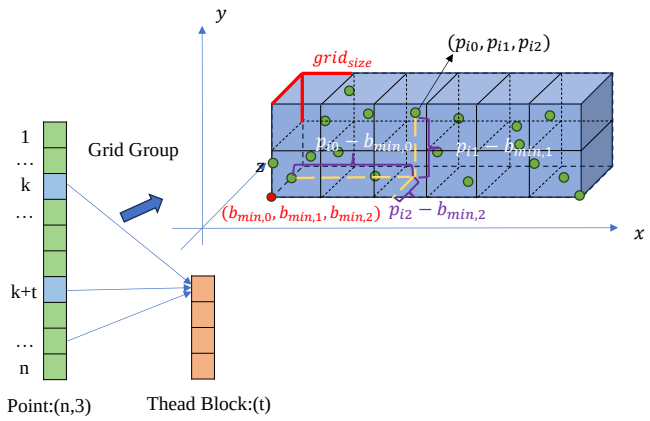

Fig. 4: Grid parallel group.

The specific strategy, as shown in Figure 4, involves obtaining the coordinates of the grid where the fruit point cloud p_i is located through the hash function in Equation (7) within the online program thread numbered k,

$$g_{ij} = \beta \left(\frac{p_{ij} - b_{min,j}}{grid_{size}} \right) \tag{7}$$

where β represents the floor function. By mapping points to their corresponding grids using grid coordinates, the time overhead caused by sorting is avoided, thereby achieving efficient grouping of fruit point clouds into grids.

B. Grid Random Sample Algorithm Based on Density

For the irregularly shaped fruit point cloud data obtained by depth cameras, which contain a large number of discrete points and holes, the point cloud density at the edges of the holes is relatively low. Along the radial line from the hole to the center of the dataset, the point cloud density increases sharply. The FPS method selects key features based on the farthest point, resulting in a concentration of feature points at the low-density edges, while the high-density regions have relatively fewer feature points. This leads to poor global coverage of the extracted key features. Additionally, the FPS method often requires extensive sorting techniques to determine the farthest distances, resulting in low efficiency.

To address the aforementioned issues, a density-based grid random sampling algorithm (GDSample) was proposed based on the fruit point cloud data after grid parallel grouping in Section 3.A. Firstly, we select key features based on density rather than distance, optimizing the uniformity of sampling for point clouds of porous fruits. Subsequently, we eliminate distance calculations and sorting through grid partitioning, while using grid random sampling to balance efficiency and accuracy. Finally, we parallelize the algorithm to make the sampling process more efficient. The specific operation of the algorithm is as follows:

First, we define the index v_i of point p_i in the grid as the number of point clouds currently held by the grid when p_i is stored. Subsequently, for a thread with the sequence number k, the sampling rate is set as shown in Equation (8):

$$r_k = \max(v_i) \times \frac{m}{n} \tag{8}$$

where n represents the total number of input point clouds, and m is the number of sampled points. The sampling rate is calculated based on the maximum density, ensuring that a greater number of key features are selected from dense point clouds.

Then, perform the operation as shown in Equation (9) on the index v_i of the fruit point p_i in the grid.

$$v_i^{'} = v_i \ \% \ r_k \tag{9}$$

Finally, obtain the fruit sampling point set as shown in Equation (10):

$$S = \left\{ s_{ej} = \Phi\left(v_i^{'}\right) \middle| \begin{matrix} e = k, k+t, \ldots, k+\mu t; \\ k + \mu t \leq m; \end{matrix} \right\} \tag{10}$$

The Φ function represents the sequential selection of points $v_i' = \{0, 1, \ldots, r_k - 1\}$ until a sufficient number of sampling points is obtained. Starting from 0, the Φ function ensures that each grid cell is sampled at least once, effectively extracting sparse edge points while capturing more features from dense point clouds, thereby achieving uniform coverage of key features. This process indirectly enhances sampling efficiency through a randomized sampling approach.

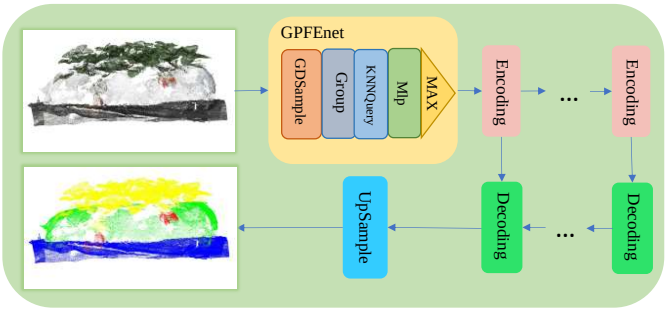

Fig. 5: Design of GPFEnet and embedding into classic models.

C. Design of GPFEnet

To maintain the accuracy of fruit point cloud semantic segmentation while improving the speed of feature extraction, we designed the GPFEnet as shown in Figure 5. The specific feature extraction process is as follows:

Sampling Based on GDSample. For input point cloud data with the shape [N, 3+c], the 3D coordinate portion [N, 3] is first extracted as input. Key feature coordinates are then extracted using GDSample, resulting in a feature output with the shape [M].

Feature Point Generation. The input point cloud [N, 3] and the extracted key feature coordinates [M] are grouped together, generating feature points with the shape [M, 3], which supports subsequent operations.

Local Feature Capture Based on KNNQuery. To more efficiently aggregate neighboring features, KNNQuery is used instead of BallQuery as the grouping metric, enhancing the ability to capture local features.

Feature Transformation Based on MLP. First, a linear layer (Linear) is used to transform the features into a new feature space. Then, to avoid gradient explosion or vanishing, LayerNorm is employed instead of BatchNorm, reducing noise introduction and promoting faster network convergence. Finally, ReLU is chosen as the activation function to mitigate the risk of overfitting while enhancing the network's robustness. Using MLP layers for complex feature extraction and nonlinear transformations allows the input features to enter a new feature space, better learning the semantic information of the features.

Feature Aggregation. The MaxPool function is used instead of AveragePool for feature aggregation. The former effectively reduces computation, captures edge features, and improves overall network efficiency.

Through the above design, GPFEnet can be more efficiently integrated into various point cloud processing networks, further enhancing network performance while maintaining computational efficiency and applicability. To strengthen GPFEnet's adaptability across different networks, we embedded the proposed GPFEnet module, as shown in Figure 5, into established network architectures, including PointNet++, PointWeb, PAConv, and Point Transformer, to validate its effectiveness and versatility in boosting the performance of these networks.

IV. Experiments

A. Agricultural Scene Datasets

1) Strawberry Dataset: To validate the uniform and efficient sampling of the GDSample algorithm on point clouds of porous fruits and the effectiveness of the GPFENet in maintaining the accuracy of fruit semantic segmentation, thereby expanding the application of point cloud deep learning in agriculture, we first chose to conduct experiments using a strawberry dataset [5] synthesized with a depth camera in complex field scenarios. Compared to other crops, strawberries have a short growth cycle, low plant stature, and dense foliage. The fruits grow among the leaves and are severely occluded, making them difficult to photograph. Therefore, in the process of capturing images containing strawberry fruits, it is inevitable to cause angular deviations between the captured images and the plants, resulting in many discrete points, holes, and uneven density in the synthesized point clouds.

Fig. 6: Strawberry dataset sample point cloud.

This dataset consists of four types of objects, including the ground of the field, the base of the strawberry, the strawberry plant, and the fruit, as detailed in Figure 6. Each set of reconstructed point clouds is very dense, comprising 30 point cloud scenes in the training-validation set and 7 point cloud scenes in the test set. Each strawberry plant contains approximately 4 to 6 million points, with specific point cloud data as shown in Table I. Comprehensive analysis indicates that selecting strawberry point cloud data with multiple holes and discrete points is appropriate for validating our network.

TABLE I: The number of strawberry point cloud (m: millions).

Dataset Category	Floor 0	Base 1	Plant 2	Fruit 3	Total
Training set	263.9m	128.6m	240.8m	16.5m	649.9m
Test set	64.5m	42.7m	42.1m	1.4m	150.8m
Total	328.4m	171.4m	282.9m	17.9m	800.8m

2) Cabbage Dataset: To evaluate the generalization performance of the proposed GPFEnet on fruit point cloud datasets, we selected the Chinese flowering cabbage dataset for testing. In field scenarios, Chinese flowering cabbage exhibits low color contrast, large leaves and branches, minimal occlusion, and ease of photography, resulting in fewer discrete points and holes during point cloud synthesis. Its fruit phenotype significantly differs from that of strawberries, as illustrated in Figure 7.

Fig. 7: Cabbage dataset sample point cloud.

This dataset comprises three types of objects: flower pots, leaves, and stems. Compared to the strawberry dataset, the reconstructed point cloud density of this dataset is relatively sparse, with each Chinese flowering cabbage plant containing approximately 300,000 points. The specific point cloud data is detailed in Table II.

TABLE II: The number of cabbage point cloud (m: millions).

Dataset Category	Flowerpot 0	Leaf 1	Branch 2	Total
Training set	20.29m	13.39m	0.68m	34.43m
Test set	8.83m	4.65m	0.17m	13.65m
Total	29.13m	18.04m	0.86m	48.09m

B. Speed Performance Validation

In this section, we systematically evaluate the efficiency of existing sampling methods, including FPS, IDIS, GS, CRS, PGS, as well as CUDA-based cudaFPS and GDSample, using the strawberry point cloud dataset. Given strawberry point clouds of varying scales $\{10^3, 5\times10^3, 10^4, 5\times10^4, 10^5\}$, we apply the aforementioned sampling methods to downsample the fruit point clouds, with a sampling rate set at $n/m = 4$. The experimental setup includes an Intel i7-12700H @ 2.3GHz CPU and an NVIDIA RTX 3050Ti GPU. The total time taken by each sampling method to process point clouds of different sizes is presented in Table III.

TABLE III: Point cloud sampling speed comparison (ms). The number of point clouds ranges from 10^3 to 10^5.

Model Category	10^3	5×10^3	10^4	5×10^4	10^5
FPS	1.17	7.34	20.13	200.29	1489.48
IDIS	4.25	19.85	39.79	101.23	399.48
GS	20.38	97.57	496.34	104	6×10^4
CRS	2.48	4.26	6.34	11.27	42.14
PGS	4.25	3.57	3.27	5.37	8.79
cudaFPS	22.52	22.86	22.88	39.26	121.99
GDSample (Ours)	0.49	0.60	0.63	1.31	3.88

In the sample speed visualization of Figure 8, the following conclusions can be drawn:

For small-scale point clouds ($\sim 10^3$), except for GS and cudaFPS, other sampling methods have similar time consumption and are unlikely to impose a heavy computational burden.

As the scale of the point cloud increases, the rate curves of FPS/IDIS/GS rise particularly sharply. Although the increase in CRS/PGS/cudaFPS is more gradual, their time consumption is still significantly higher than that of GPFEnet. By comparing

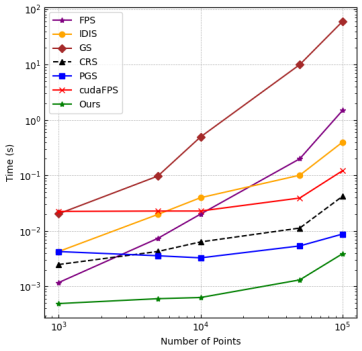

Fig. 8: Time consumption of different sampling approaches.

the time consumption of CUDA-based cudaFPS and GDSample, we find that the latter has a CUDA acceleration ratio of approximately 35 times compared to the former.

From the above analysis, it can be concluded that the proposed algorithm has significantly lower time consumption than traditional, more expensive sampling methods, and the sampling speed is significantly improved, demonstrating the efficiency of the GDSample algorithm in sampling on the porous strawberry dataset, thereby significantly improving feature extraction efficiency.

C. GDSample Sampling Uniformity Validation

To further validate the effectiveness of GDSample in optimizing sampling uniformity, we selected a strawberry dataset with multiple cavities and discrete points for visualization of sampling results. Specifically, different sampling methods, FPS and GDSample, were employed to extract key features at varying sampling rates of $n/m = 4096/m = \{2, 4, 8, \ldots, 1024\}$, The sampled point sets were then visualized as shown in Figure 9, where the blue point clouds represent the key features extracted by the sampling algorithms.

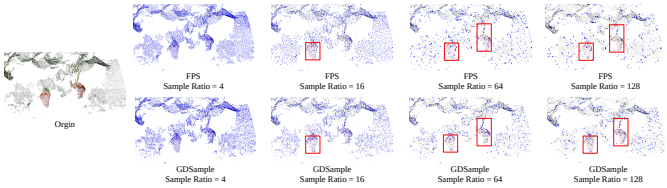

Fig. 9: Visualization of feature points, using FPS and GDSample on the strawberry dataset at different sampling rates.

By analyzing Figure 9, the following observations can be made:

When the sampling rate is 4, the distribution of feature points extracted by GDSample and FPS shows minimal differences.

When the sampling rate is 16, FPS primarily extracts key features concentrated at the edge points, whereas GDSample not only extracts more features in high-density regions but also effectively captures edge points of the point cloud.

TABLE IV: The segmentation accuracy on strawberry dataset.

Model Category	mIoU(%)	mAcc(%)	allAcc(%)	Floor(%)	Base(%)	Plant(%)	Fruit(%)
Pointnet2	51.95	66.17	75.95	66.61	41.58	64.09	35.51
PointWeb	74.28	80.85	92.49	89.26	73.07	90.64	44.17
PAConv	75.42	84.83	91.62	87.52	67.38	91.96	54.83
Point Transformer	68.16	79.39	83.95	69.66	54.20	90.03	58.73
Pointnet2 + GPFEnet	**54.41**	**69.61**	**79.55**	**69.24**	33.50	**74.48**	**40.43**
PointWeb + GPFEnet	**75.65**	**83.28**	**93.60**	87.18	69.46	**90.76**	**55.22**
PAConv + GPFEnet	**78.09**	**86.61**	**92.65**	**89.39**	71.82	**91.49**	**59.65**
Point Transformer + GPFEnet	**70.33**	**79.66**	**86.22**	**78.64**	**59.17**	89.53	53.96

TABLE V: The segmentation accuracy on cabbage dataset.

Model Category	mIoU(%)	mAcc(%)	allAcc(%)	Flowerpot(%)	Leaf(%)	Branch(%)
Pointnet2	53.40	67.92	98.17	97.70	91.91	24.00
PointWeb	54.63	68.72	98.08	99.54	90.48	23.80
PAConv	64.58	69.24	98.67	98.50	95.91	63.93
Point Transformer	67.77	70.02	98.77	98.47	96.69	75.93
Pointnet2 + GPFEnet	**57.34**	**68.74**	98.04	**97.92**	**92.91**	**38.52**
PointWeb + GPFEnet	53.75	**69.76**	97.94	98.73	**91.13**	**25.14**
PAConv + GPFEnet	63.19	69.03	98.50	98.38	95.78	58.62
Point Transformer + GPFEnet	67.34	69.49	**99.14**	**99.08**	**97.83**	72.44

As the sampling rate further increases to 64 and 128, the advantages of the proposed algorithm become particularly pronounced. FPS mainly extracts feature points at the edges of the point cloud, leading to insufficient coverage of features in the central region. In contrast, GDSample not only fully retains the features of the edge regions but also achieves uniform coverage in the central part of the point cloud, optimizing the uniformity of sampling.

These findings validate the superiority and robustness of GDSample in sampling porous point clouds in complex field scenarios, demonstrating its adaptability and global feature coverage capabilities at different sampling rates.

D. GPFEnet Fruit Dataset Segmentation Experiment

In this section, we systematically evaluate the accuracy of GPFEnet for point cloud semantic segmentation in complex field scenarios. Specifically, using strawberry and Chinese cabbage field datasets as input point clouds, the proposed GPFEnet module is embedded into classic network architectures mentioned before.

The specific experimental parameters are set as follows: normalized color is concatenated with xyz coordinates as input data, the sampling rate is set to $n/m = 4$, the SGD optimizer with momentum is used, with momentum and weight decay set to 0.9 and 0.0001 respectively, the initial learning rate is set to 0.55, training is conducted for 200 epochs, and the experimental setup includes an Intel Xeon Platinum 8255C @ 2.5GHz CPU and an NVIDIA RTX 3080Ti GPU.

1) Strawberry Segmentation: To validate the segmentation accuracy of GPFEnet on point clouds of porous fruits, we initially conducted experiments embedding GPFEnet into different networks on a strawberry dataset. The evaluation metrics for the experiments included mIoU, mean Accuracy (mAcc), overall point Accuracy (OA), and the mean Intersection over Union(Iou) for the four categories of strawberries (including floor, base, leaves, and fruit). The segmentation results of the strawberry dataset are visualized in Figure 10.

Accuracy Analysis. As shown in Table IV. the mIoU of PointNet++ increased by 2.46 % after embedding the GPFEnet module. For PointWeb, PAConv, and Point Transformer, the mIoU increased by 1.37%, 2.67%, and 2.17%, respectively. For the floor, base, leaves, and fruits of strawberries, most IoUs have improved to some extent, indicating that the GPFEnet is more accurate in segmenting porous strawberry data compared to the original model.

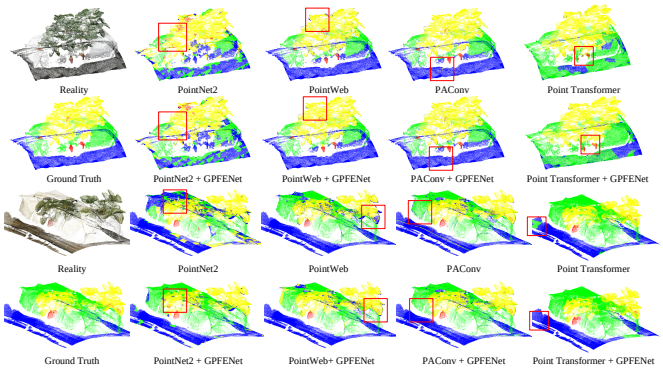

Fig. 10: Visual comparison on strawberry dataset.

From the above analysis, it can be concluded that the model embedded with GPFEnet alleviates the impact of porous data synthesized by depth cameras on segmentation accuracy and more uniformly extracts key features in complex scenes.

2) Cabbage Segmentation: To further validate the effectiveness of GPFEnet in fruit segmentation, we selected cabbage dataset with fewer holes and discrete points for generalization experiments. The evaluation metrics for the experiments included the mIou, mAcc, OA, and the Iou for the three categories of Chinese flowering cabbage (including flowerpot, stem, and leaf). The segmentation visualization of Chinese flowering cabbage is shown in Figure 11.

Accuracy Analysis. The experimental accuracy of the Chi-

nese flowering cabbage is shown in Table V. All three categories of Chinese flowering cabbage showed improvements on Pointnet++, with mIoU increasing by 3.94% and mAcc by 0.82%. In the PAConv experiments, all categories except for the flowerpot showed improvements. On Point Transformer, allAcc increased by 0.37%, with all accuracies except for the leaf showing improvements. However, the segmentation mIoU on PAConv decreased by 1.39%. Through analysis, we believe that although the random sampling in the GDSample algorithm led to a slight decrease in segmentation accuracy, it significantly improved the processing speed.

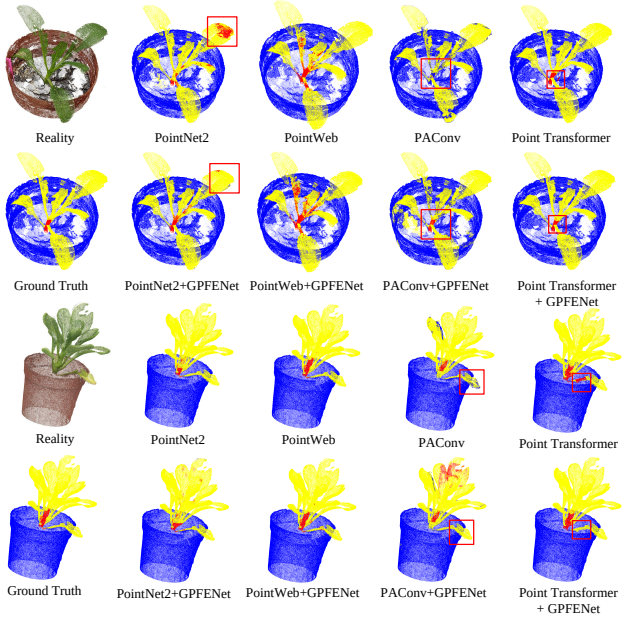

Fig. 11: Visual comparison on cabbage dataset.

From the above analysis, it can be concluded that some models embedded with the GPFEnet have achieved a certain degree of improvement on the cabbage dataset with fewer holes. By comparing the segmentation effects with the strawberry dataset, it is evident that the proposed network is more suitable for datasets with more holes, demonstrating the network's superiority in segmenting porous datasets in complex field scenarios.

V. Conclusion

To optimize the sampling uniformity of point clouds with multiple holes in agricultural field scenes while enhancing feature extraction efficiency, this paper presents the GDSample algorithm, a lightweight and efficient density-based grid parallel sampling method, alongside GPFEnet, a grid parallel feature extraction network. Experiments on the strawberry dataset demonstrate that GDSample is approximately 35 times faster than FPS, significantly improving sampling efficiency for porous point clouds. The algorithm also effectively captures global features, ensuring uniformity in sampling. Segmentation tests on both the strawberry and Chinese flowering cabbage datasets show that GPFEnet, while maintaining speed, achieves competitive segmentation accuracy, with improvements of 2.67% mIoU on strawberries and 3.94% on bok choy. These results highlight the method's ability to balance efficiency and accuracy, offering a viable solution to the challenges of feature extraction and coverage in complex agricultural environments with porous, discrete point clouds.

While random sampling increases efficiency, it may slightly reduce accuracy on some models compared to FPS. Future work will focus on optimizing the model, increasing accuracy to better accommodate lightweight agricultural robots and similar applications.

References

[1] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, "Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 632–11 641.

[2] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers, "D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1281–1292.

[3] Z. H, J. L, J. J, and et al., "Point transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 259–16 268.

[4] Y. Li, W. Wen, T. Miao, S. Wu, Z. Yu, X. Wang, X. Guo, and C. Zhao, "Automatic organ-level point cloud segmentation of maize shoots by integrating high-throughput data acquisition and deep learning," *Computers and Electronics in Agriculture*, vol. 193, p. 106702, 2022.

[5] Q. Liu, H. Yang, J. Wei, Y. Zhang, and S. Yang, "Fsdnet: A features spreading net with density for 3d segmentation in agriculture," *Computers and Electronics in Agriculture*, vol. 222, p. 109073, 2024.

[6] Q. C. R., Y. L., S. H., and et al., "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[7] L. X, L. J, J. L, and et al., "Stratified transformer for 3d point cloud segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8500–8509.

[8] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, "Pointweb: Enhancing local neighborhood features for point cloud processing," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5565–5573.

[9] X. M, D. R, Z. H, and et al., "Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3173–3182.

[10] Y. Chen, Y. Xiong, B. Zhang, J. Zhou, and Q. Zhang, "3d point cloud semantic segmentation toward large-scale unstructured agricultural scene classification," *Computers and Electronics in Agriculture*, vol. 190, p. 106445, 2021.

[11] R. Jayakumari, R. R. Nidamanuri, and A. M. Ramiya, "Object-level classification of vegetable crops in 3d lidar point cloud using deep learning convolutional neural networks," *Precision Agriculture*, vol. 22, no. 5, pp. 1617–1633, 2021.

[12] F. Hu, C. Lin, J. Han, and J. Peng, "Sementing the field of rapeseed from 3d laser point cloud using deep learning," in *2021 9th International Conference on Agro-Geoinformatics (Agro-Geoinformatics)*. IEEE, 2021, pp. 1–5.

[13] Q. Yu, H. Yang, Y. Gao, X. Ma, G. Chen, and X. Wang, "Lfpnet: Lightweight network on real point sets for fruit classification and segmentation," *Computers and Electronics in Agriculture*, vol. 194, p. 106691, 2022.

[14] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[15] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 697–12 705.

[16] C. Yilun, L. Shu, S. Xiaoyong, and J. Jiaya, "Fast point r-cnn," in *ICCV*, 2019.
[17] B. Graham, M. Engelcke, and L. Van Der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9224–9232.
[18] T. Le and Y. Duan, "Pointgrid: A deep network for 3d shape understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9204–9214.
[19] O. Dovrat, I. Lang, and S. Avidan, "Learning to sample," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2760–2769.
[20] I. Lang, A. Manor, and S. Avidan, "Samplenet: Differentiable point cloud sampling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7578–7588.
[21] A. Abid, M. F. Balin, and J. Zou, "Concrete autoencoders for differentiable feature selection and reconstruction," *arXiv preprint arXiv:1901.09346*, 2019.
[22] J. Yang, Q. Zhang, B. Ni, L. Li, J. Liu, M. Zhou, and Q. Tian, "Modeling point clouds with self-attention and gumbel subset sampling," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3323–3332.
[23] E. Nezhadarya, E. Taghavi, R. Razani, B. Liu, and J. Luo, "Adaptive hierarchical down-sampling for point cloud classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 956–12 964.
[24] Y. Qian, J. Hou, Q. Zhang, Y. Zeng, S. Kwong, and Y. He, "Mops-net: A matrix optimization-driven network fortask-oriented 3d point cloud downsampling," *arXiv preprint arXiv:2005.00383*, 2020.
[25] Y. Lin, Y. Huang, S. Zhou, M. Jiang, T. Wang, and Y. Lei, "Da-net: density-adaptive downsampling network for point cloud classification via end-to-end learning," in *2021 4th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*. IEEE, 2021, pp. 13–18.
[26] L.-Y. Wei, "Parallel poisson disk sampling," *Acm Transactions On Graphics (tog)*, vol. 27, no. 3, pp. 1–9, 2008.
[27] S. T. Tokdar and R. E. Kass, "Importance sampling: a review," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 1, pp. 54–60, 2010.
[28] A. Gunawan and M. de Berg, "A faster algorithm for dbscan," *Master's thesis*, 2013.
[29] C. Deng, J. Song, R. Sun, S. Cai, and Y. Shi, "Griden: An effective grid-based and density-based spatial clustering algorithm to support parallel computing," *Pattern Recognition Letters*, vol. 109, pp. 81–88, 2018.