Mathematical Biosciences and Engineering

MBE, 21(3): 4669–4697.
DOI: 10.3934/mbe.2024205
Received: 12 December 2023
Revised: 07 February 2024
Accepted: 08 February 2024
Published: 29 February 2024

http://www.aimspress.com/journal/MBE

Research article

The improved stratified transformer for organ segmentation of Arabidopsis

Yuhui Zheng[1], Dongwei Wang[1], Ning Jin[2], Xueguan Zhao[3], Fengmei Li[4], Fengbo Sun[5], Gang Dou[6] and Haoran Bai[1,*]

[1] College of Mechanical and Electrical Engineering, Qingdao Agricultural University, Qingdao 266109, China
[2] Graduate School, Shenyang Jianzhu University, Shenyang 110168, China
[3] Beijing PAIDE Science and Technology Development Co., Ltd., Beijing 100097, China
[4] College of Food Science and Engineering, Qingdao Agricultural University, Qingdao 266109, China
[5] China Zhongxin Construction Engineering Co., Ltd., Qingdao 266205, China
[6] Weichai Lovol Intelligent Agricultural Technology Co., Ltd., Weifang 261000, China

* **Correspondence:** Email: baihaoran111@126.com; Tel: +8613854285625.

Abstract: Segmenting plant organs is a crucial step in extracting plant phenotypes. Despite the advancements in point-based neural networks, the field of plant point cloud segmentation suffers from a lack of adequate datasets. In this study, we addressed this issue by generating Arabidopsis models using L-system and proposing the surface-weighted sampling method. This approach enables automated point sampling and annotation, resulting in fully annotated point clouds. To create the Arabidopsis dataset, we employed Voxel Centroid Sampling and Random Sampling as point cloud downsampling methods, effectively reducing the number of points. To enhance the efficiency of semantic segmentation in plant point clouds, we introduced the Plant Stratified Transformer. This network is an improved version of the Stratified Transformer, incorporating the Fast Downsample Layer. Our improved network underwent training and testing on our dataset, and we compared its performance with PointNet++, PAConv, and the original Stratified Transformer network. For semantic segmentation, our improved network achieved mean Precision, Recall, F1-score and IoU of 84.20, 83.03, 83.61 and 73.11%, respectively. It outperformed PointNet++ and PAConv and performed similarly to the original network. Regarding efficiency, the training time and inference time were 714.3 and 597.9 ms, respectively, which were reduced by 320.9 and 271.8 ms, respectively, compared to the original network. The improved network significantly accelerated the speed of feeding point clouds into the network while maintaining segmentation performance. We demonstrated the potential of virtual plants and deep learning methods in rapidly extracting plant phenotypes,

contributing to the advancement of plant phenotype research.

Keywords: virtual plant; plant phenotyping; deep learning; point cloud; plant segmentation

1. Introduction

Plant phenotypes refer to the morphological characteristics controlled by genes in individual or populations of plants under specific environments, which is a comprehensive reflection of plant traits [1]. The major components of plant phenotype include organs, such as roots, leaves, stems, and fruits, with leaves being the most common phenotypic trait. Traditional measurement methods rely on manual observation and are subject to subjective errors. This makes it difficult to meet the demand for rapid phenotypic analysis of large-scale plants [2]. In the past decade, automated phenotype research using computer vision and machine learning techniques has received widespread attention [3]. The core issue lies in how to effectively and accurately segment plant organs.

In recent years, advanced deep learning methods based on 2D images have been widely used for plant part segmentation [4], such as the semantic segmentation residual U-Net model for plant images [5], and methods for leaf segmentation used in disease identification [6–9]. However, the specific structure of plants and the influence of factors like complex environmental conditions during plant growth limit the accuracy of obtaining precise plant phenotypic parameters through 2D images. 2D images lack depth information, leading to issues such as occlusion, data loss [10], and variability caused by lighting conditions [11].

3D point cloud models can provide accurate spatial information of objects [12] including their position, shape, size and orientation. In contrast, 2D images can only provide the projected information of objects on the plane and cannot accurately capture their three-dimensional features. Machine vision-based depth cameras and binocular vision-based 3D imaging techniques have been used for 3D reconstruction of small plants, such as soybeans [13] and corn [14,15]. Compared to machine vision-based 3D imaging techniques, LiDAR is more accurate and is commonly used for large-scale scenes such as forests [16] and farmlands [17,18]. However, LiDAR data processing requires significant workload, time, and cost. Compared to the two methods mentioned above for obtaining 3D point cloud models of plants, synthetic 3D virtual plant models have the advantage of low cost and simple acquisition methods. Synthetic plant models have long been used in plant research to simulate the interaction between plants and the environment [19]. Lindenmayer systems (L-systems) [20] is a type of string rewriting system that can be used to generate fractals and natural patterns. Platforms for constructing plant models are typically developed based on L-systems, such as the L + C modeling language combined with C language [21] and the L-Py framework combined with Python [22]. Some studies utilize synthetic plants as training data [23]. Reference [24] establishes models capable of automatically annotating maize and canola, saving the time required for manual labeling. Similarly, in [25], L-Systems are employed to model artichoke seedlings and automate annotation, thereby reducing cost and time. The study [26] combines real and synthetic images of Arabidopsis for network training. Reference [27] uses 3D plant models generated by Blender and scenes with random plant parameters to create a synthetic dataset. Reference [28] shows that using high-quality 3D synthetic plants to augment the dataset improves performance in leaf counting tasks. All of the above involve using synthetic 3D plant models to generate 2D images rather than directly incorporating synthetic 3D

models into training. In contrast, references [29,30] use synthetic roses and Arabidopsis, respectively, as substitutes for real plants directly involved in deep learning training.

Point clouds are composed of a large number of discrete points, exhibiting disorder and irregularity, making segmentation with deep learning methods challenging [31]. For handling irregular inputs like point clouds, an intuitive approach is to transform the irregularity into regularity. One approach is multi-view projection [32], which projects 3D point cloud information onto a 2D plane. Shi et al. [33] employed the multi-view projection method for semantic and instance segmentation of tomato seedlings. However, in the multi-view projection method [34], the geometric information of point clouds is collapsed during the projection stage. Another approach to transforming irregular point clouds into regular representations is 3D voxelization [35], followed by using deep learning to process plant point clouds [36]. Compared to the two aforementioned indirect methods, directly point-based deep learning networks consume less computation and memory. PointNet and PointNet++ [37,38] are the earliest point-based 3D deep neural networks. Kang et al. [39] and Masuda [40] applied these two networks to plant point cloud segmentation. Subsequently, point-based 3D neural networks specialized in plant point cloud segmentation were proposed [41]. Ghahremani et al. [42] proposed a lightweight deep network for plant point cloud segmentation, and later introduced Pattern-Net [43], specifically for wheat point cloud segmentation, based on the original network. Recently, attention-based Transformers have been first applied to point-based deep learning [44–46], significantly improving the performance of point cloud segmentation. Subsequently, some point-based deep learning network models combined with Transformers have been proposed [47–49], demonstrating excellent segmentation performance. Turgut et al. [29] and Li et al. [50] proposed attention-based point cloud segmentation networks, RoseSegNet and PSegNet, specifically designed for plant point cloud segmentation.

In conclusion, using 3D deep learning for plant organ segmentation is a promising method in phenotypic research, but there are several challenges. Due to the lack of plant datasets, it is challenging to validate the effectiveness of the designed networks on complex plant structures. In most 3D deep learning models, methods such as downsampling and grouping points may incur significant time costs [51].

To address the aforementioned challenges, we created virtual Arabidopsis models that simulate the characteristics of real Arabidopsis complex structures, which can be used to verify the feasibility of our network for segmentation on plants with complex structures. We proposed a novel downsampling module to improve the Stratified Transformer network [52] for semantic segmentation on our created virtual Arabidopsis dataset. Our network consumed less time compared to the original network while ensuring segmentation performance.

Our major contributions were as follows:

- To obtain more Arabidopsis data, we used L-Py to generate virtual Arabidopsis models. Then, we proposed the surface-weighted sampling method and downsampling method to obtain annotated Arabidopsis point clouds from the virtual models.
- Plant Stratified Transformer was proposed by us. This network utilized the Fast Downsample Layer, replacing the Stratified Transformer downsampling layer. Testing on the Arabidopsis dataset demonstrated that the improved network maintained segmentation performance comparable to the original network while reducing both training and inference times.
- Virtual Arabidopsis was involved in network training to validate the semantic segmentation performance of different networks, tested against real Arabidopsis. This demonstrated that when there was only a small amount of real plant data, virtual plants could be used as training data for the network.

2. Materials and methods

2.1. Virtual Arabidopsis model

Virtual plants are computer-generated synthetic models that simulate the development and growth processes of real plants [28]. L-systems is a commonly used formalism that can describe a wide range of plant features and types [53]. In the L-system framework, plants are defined by a sequence of symbols referred to as an L-string, which represents various organs of the plant. We utilized L-Py, a Python-based L-system, as the development platform [22]. On this platform, we employed the algorithm proposed by Chaudhury et al. [54] for plant generation to create Arabidopsis models. As shown in Figure 1(a), the generation of the Arabidopsis model primarily involves the following L-strings:

- $Meristem(t)$ for apical or lateral meristems.
- $Leaf(t, label\_L)$ for leaves of different sizes.
- $Internode(t, label\_I)$ for stem.
- $Flower(t, label\_F)$ for flowers composed of a pedicel, petals, and carpels, with the carpel subsequently developing into a fruit.

Each symbol was associated with a set of parameters that described the variables attached to the respective organ. The definitions of organs in Figure 1(a) essentially represent recursive functions, and thus, the generation and evolution of each organ involve iterative recursion with respect to the corresponding symbol over time increments of *dt* (1 hour). During the generation of organs, each class of organs X that needs to be identified is assigned a unique 'label_X'.

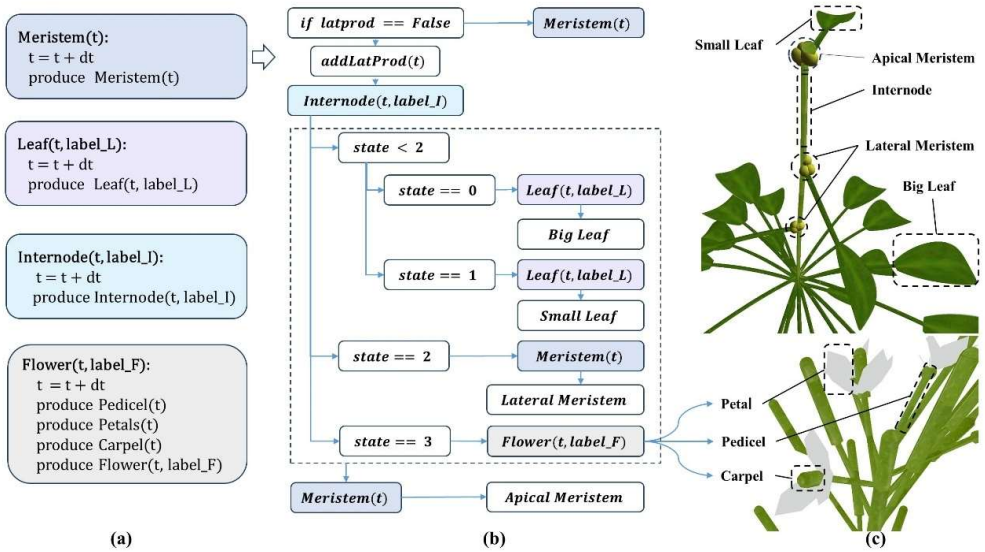

Figure 1. The generation process of virtual Arabidopsis. (a) represents the definitions of major plant organs, (b) illustrates the process of Meristem developing into different organs, and (c) shows the morphologies of different organs during the development process.

To ensure the reality of virtual plants, just like real plant development, all organs originated from the Meristem. The morphologies of organs during growth and development are shown in Figure 1(c). Therefore, the generation process of Arabidopsis could be described simply as the development of

different organs from Meristem under different growth states, as shown in Figure 1(b). The development of plant organs in the figure depends on the value of 'state', which is provided by $addLatProd(t)$. In order to explain the process of organ development in more detail, we presented the pseudocode (Algorithm 1) for this process. The pseudocode contains a lot of descriptive language for ease of understanding.

Algorithm 1: The process of Meristem developing into different organs.

```
    Input: Growth time t
    Output: Internode(t) ,leaf , flower and Meristem(t)
1   if latprod == False:
2       //a plastochrone time is not reached so a lateral production(latprod) is not produced
3       Meristem(t): continue iteration
4   else:
5       //a plastochrone is reached and a lateral production must be added
6       addLatProd(t):
7           Physiological state [s, d]:
8               s = 0: Vegetative state
9               s = 1: Between Vegetative and In flowering state
10              s = 2: In flowering state
11              s = 3: Flower state
12              d = Plastochrones in this state (duration)
13          if a lateral primordium is produced:
14              Update of state counters d for the current meristem
15          //Check physiological age
16          if duration limit is reached:
17              move to the next state s
18      produce:
19          Internode(t)
20          A Meristem in state [s, d]:
21              if d < d_s (duration threshold in state s):
22                  produce a lateral meristem in state [s + 1, 0]
23                  produce an apical meristem in state [s, d + 1]
24              if d = d_s:
25                  produce a lateral meristem in state [s + 2, 0]
26                  produce an apical meristem in state [s + 1, 0]
27              In addition: produce leaf in state 0 and 1
28              Finally: the meristem transforms into a flower in state 3
29          Meristem(t): keep an apical growth
```

Figure 2(d),(e) [55] depict the comparison between the generated Arabidopsis structure and the real Arabidopsis. In order to ensure the randomness of plant growth while preserving the structure imitating real plants, we employed a significant amount of random computation. Typically, we achieved this by assigning two values to a variable, one as the mean and the other as the standard deviation, and during the runtime, we randomly generated values by substituting these two values into

a Gaussian distribution. For instance, the growth duration (in hours) was determined by providing us with the mean days (Mean_day) and standard deviation (StDev_day), and the final duration in days was obtained by the Gaussian distribution function, which was then multiplied by 24. The leaf morphology was determined by a randomly computed ratio between the leaf blade and the petiole. These methods ensure that variables can fluctuate within certain constraints, thus maintaining both developmental diversity and the stability of the plant.

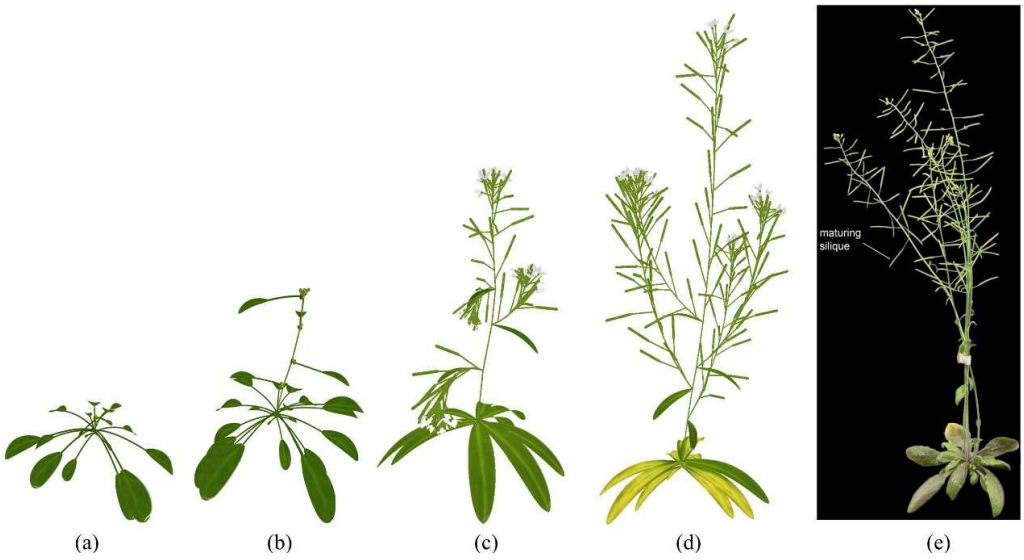

Figure 2. Virtual Arabidopsis at different time points. The average growth periods for (a), (b), (c) and (d) are 10, 20, 30 and 40 days, respectively. (e) represents real Arabidopsis with a growth period of 59 days [55].

The Arabidopsis we generated using the above algorithm is shown in Figure 3(a). The generated Arabidopsis consists of a large number of triangles meshes of different sizes, as shown in Figure 3(b). The simplest method to convert the mesh model into a point cloud was to directly use the intersection points in the mesh as sampling points, but this results in significant shape errors in the point cloud. Alternatively, we sampled one point from each triangle repeatedly until the specified number of points was obtained. However, because there were more small triangles than large triangles, the point cloud had highly uneven point density. Moreover, the method was inefficient since the time complexity of randomly selecting triangles each time was $O_{(N)}$, where N was the number of triangles. To address these issues, we introduced the surface-weighted sampling method, where triangles were selected based on their weight, and points on their surfaces were randomly sampled.

First, we utilized the $Alias\ Method$ [56], a non-uniform random sampling algorithm that trades space for time. The triangle area as the weight to sample one triangle at a time. The sampling process was mainly as follows: For a grid with N triangles, $Alias\ Method$ compressed the entire probability distribution into a $1 \times N$ rectangle, where each event i (corresponding to different-sized triangles) was transformed into the area it occupies within the rectangle. The area occupied by each event in the rectangle can be expressed as follows:

$$S_i = \frac{p_i \times N}{\sum_{k=1}^{k-N} p_k}. \tag{1}$$

The area occupied by event i in the rectangle may have cases where $S_i > 1$ or $S_i < 1$. We supplemented the excess area of events with $S_i > 1$ to the corresponding event with $S_i < 1$, forming small rectangles with $S_i = 1$. Additionally, we ensured that each small rectangle stores a maximum of two events. Lists $Prob$ and $Alias$ were constructed, where $Prob[i]$ represented the area proportion of event i in the i-th small rectangle, representing the probability of event i. $Alias[i]$ indicated the index of another event in the same small rectangle. This preprocessing step generated a list with a time complexity of $O_{(N)}$ and a space complexity of $O_{(N)}$. When selecting a triangle, we generated a random number $i \in [0, N)$ and then generated another random number $r \sim Unif(0,1)$. If $r < Prob[i]$, it meant event i was accepted; otherwise, event i was rejected and $Alias[i]$ was returned. The time complexity here was only $O_{(1)}$. In summary, for a mesh of N triangles, $Alias\ Method$ could pre-generate a list with a time complexity of $O_{(N)}$ and a space complexity of $O_{(N)}$, and then randomly selected triangles in constant time ($O_{(1)}$).

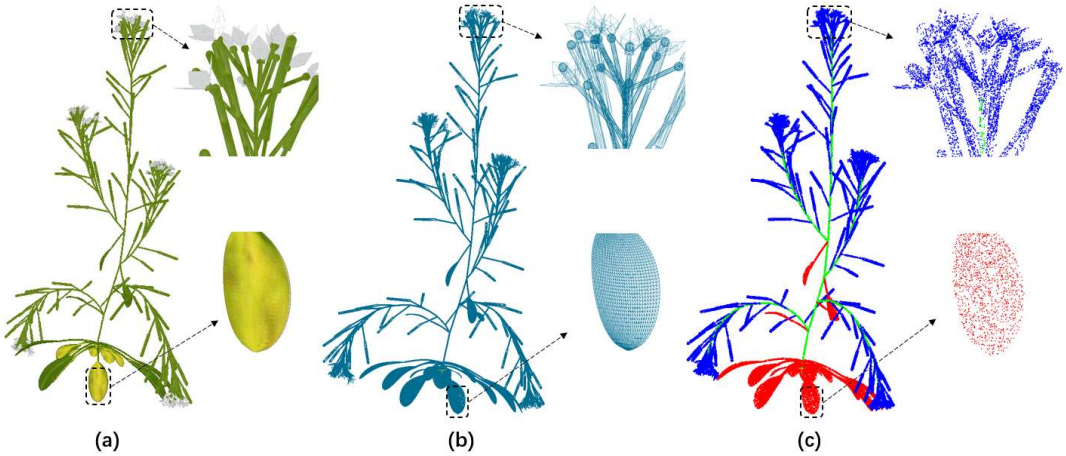

Figure 3. Virtual Arabidopsis models transforms into point cloud models. (a) is Arabidopsis original model, (b) is triangle mesh model, (c) is fully annotated point cloud model.

Then, we performed point sampling inside the selected triangle. For each selected $\triangle ABC$, we generated two random numbers u and v on $(0,1)$ and obtained the surface sampling point $P_{(x,y)}$ using Eq (2).

$$\begin{cases} x = (1-\sqrt{v})x_a + \sqrt{v}(1-u)x_b + \sqrt{v}ux_c \\ y = (1-\sqrt{v})y_a + \sqrt{v}(1-u)y_b + \sqrt{v}uy_c \end{cases} u, \mathrm{v} \in (0,1). \tag{2}$$

x_a, x_b, x_c and y_a, y_b, y_c represent the x and y coordinates of vertices A, B, C, respectively. $\sqrt{v}$ represents the percentage from vertex A to the opposite edge BC, while u represents the percentage along the edge BC (see Figure 4).

In summary, we decomposed the original model's surface into numerous triangles of varying sizes. Next, we employed the surface-weighted sampling method to repeatedly select triangles and sample random points within each selected triangle, resulting in the final point cloud. The resulting point cloud is depicted in Figure 5(c), where points of different colors correspond to different labels.

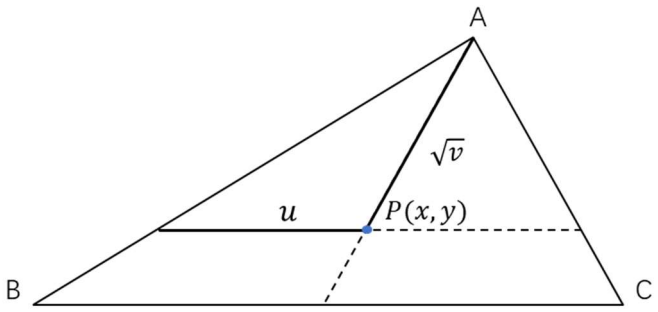

Figure 4. Sampling a random point in a triangle.

2.2. Point cloud sampling

Since neural networks have a small and fixed number of points fed during training, after obtaining the virtual Arabidopsis point cloud, downsampling is required to reduce the number of points to a fixed value. In point cloud downsampling, it is necessary to rapidly and significantly reduce the number of points while preserving the geometric shape and structural information of the plant as much as possible. In order to rapidly reduce the large and dense point cloud to a specified size, Voxel Centroid Sampling and Random Sampling methods are commonly used. Voxel Centroid Sampling first partitions the point cloud into voxels, and each voxel is sampled with only one centroid point [57]. Therefore, it can preserve the geometric information of the point cloud and reduce the existence of redundant data, but the number of points after sampling is uncertain. Random Sampling randomly samples points with the same probability, ensuring a more uniform distribution of points in the point cloud, and the number of sampled points can be specified [58]. However, during the sampling process, geometric information may be overlooked, which may lead to distortion in the sampling results.

In order to preserve the geometric information of point clouds while controlling the number of points after sampling, we combined these two sampling methods in the downsampling process, as shown in Figure 5. We define the point cloud as $\mathcal{P} = \{p_i | i = 1, \ldots, n\}$, where $p_i = (x_i, y_i, z_i)$.

For each point, we used K-Dimensional Tree to find its nearest neighbors and calculate the Euclidean distance between p_i and its nearest neighbor p_i'. Then, we average the distances of all points to obtain the average point spacing of the entire point cloud.

$$space_{mean} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(x_i - x_i')^2 + (y_i - y_i')^2 + (z_i - z_i')^2}\,. \tag{3}$$

Next, we computed the size of the voxels in the point cloud. The $size = \frac{space_{mean}}{\sqrt[d]{r}}$, where d is the dimension of the point cloud, and r is a scaling factor usually ranging from 0.5 to 0.8. The point cloud was divided into voxels of the selected size, and the centroid of each voxel was calculated as $p_{cent}\left(\frac{1}{n}\sum_{i=1}^{n} x_i\,,\ \frac{1}{n}\sum_{i=1}^{n} y_i\,,\ \frac{1}{n}\sum_{i=1}^{n} z_i\right)$. The centroids of all voxels were output as the new point cloud in Figure 5(b). Finally, the new point cloud was sampled randomly to output the specified number of points in Figure 5(c). To meet the limitations on the input point cloud size for point-based point cloud segmentation networks, the number of points in the final downsampled point cloud was set to 8192.

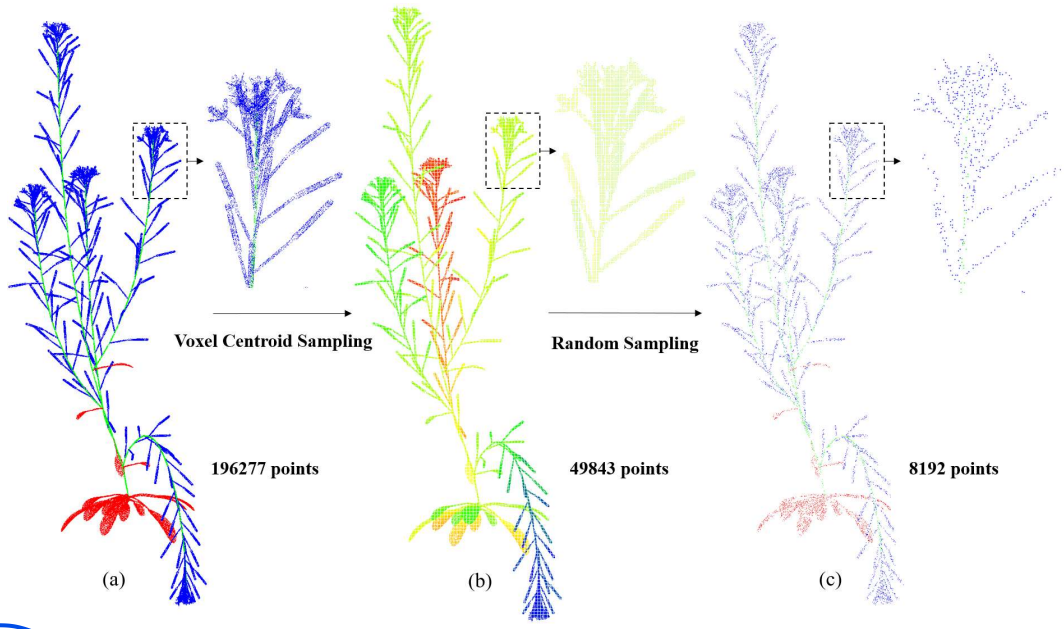

Figure 5. The downsampling process. Voxel Centroid Sampling was performed on the original point cloud (a) to obtain the voxelized point cloud (b). After sampling a fixed number of points through Random Sampling, the sampling point cloud (c) is finally obtained.

2.3. Network architecture

We improved the state-of-the-art method Stratified Transformer in the field of point cloud semantic segmentation and proposed the Plant Stratified Transformer for semantic segmentation of plant point clouds. We designed the Fast Downsample Layer, utilizing point sampling from PSNet [59] and feature normalization from Pre-LN [60] to accelerate the downsampling process.

2.3.1. Stratified transformer

Stratified Transformer [52] is a novel point-based 3D point cloud segmentation network that effectively captures long-range contextual information and exhibits strong generalization capability and high performance. To accelerate convergence speed and enhance performance, Stratified Transformer utilizes First-layer Point Embedding to aggregate local information. The greatest advantage of this network lies in its novel key sampling strategy, the Stratified Key-sampling Strategy. For each query point, it densely samples nearby points and sparsely samples distant points in a stratified manner as its keys, enabling the model to expand the effective receptive field and acquire long-range context at a lower computational cost.

The Stratified Transformer mainly consists of the First-layer Point Embedding, Transformer Block, Downsample Layers, and Upsample Layers. The Downsample Layer first performs FPS (Farthest Point Sampling) to obtain the sampled points and then uses kNN (k-Nearest Neighbors) to query the original points for grouping indices. However, the computational cost of FPS significantly increases with the number of points [61]. The sampling results of FPS are influenced by the initial point selection during the sampling process and the order of points in the point cloud. In addition, FPS only considers the Euclidean distance between points and does not take into account the local features

or global structure of the point cloud. kNN requires computing the Euclidean distance between each sampled point and all other points, but it has a high time and space complexity [62].

2.3.2. Fast downsample layer

We proposed the Fast Downsample Layer as a replacement for the original Downsample Layer. PSNet replaced FPS and kNN in Downsample Layer to perform sampling and grouping tasks, reducing computation and improving speed. Additionally, Pre-LN normalized features, and Max Pooling aggregated the projected features through grouped indexing. The entire structure of the downsampling layer is illustrated in Figure 6, where the input consists of points and features, and the output consists of sampled points and features.

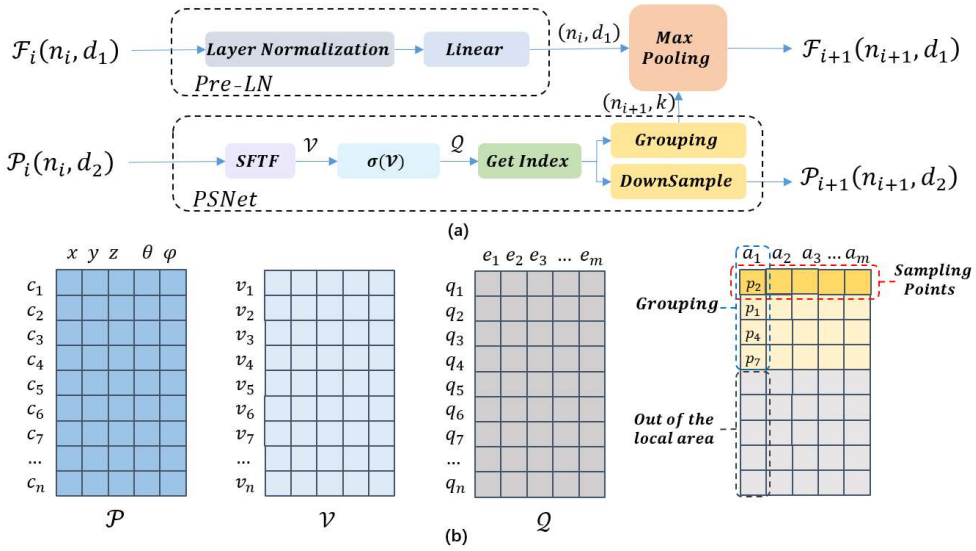

Figure 6. The structure of the downsampling layer. (a) represents the processing flow of points and point features in the downsampling. n_i, n_{i+1} indicate the number of points, d_1, d_2, k denote the dimensions. (b) is a schematic diagram of the points data structure in each step of PSNet. In the (b) rightmost image, the gray areas in each column represent points outside the grouping, the light-yellow areas represent points within the same group, and the top dark yellow area indicates the sampled points.

PSNet performs simultaneous sampling and grouping, illustrated in Figure 6(a). In Figure 6(b), $\mathcal{P}$ is the input point cloud with n points, represented as $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$. Each point p_i has spatial coordinates $c_i = (x_i, y_i, z_i, \theta_i, \varphi_i)$, where $c_i \in \mathbb{R}^d$. θ_i and φ_i are the polar and azimuthal angles, respectively, of each point in spherical coordinates. Their values can be calculated using the following formulas:

$$\begin{cases} \theta = arctan\left(\frac{\sqrt{x^2+y^2}}{z}\right) \\ \varphi = arctan\left(\frac{y}{x}\right) \end{cases}. \tag{4}$$

The point cloud $\mathcal{P}$ is divided into multiple sub-point clouds, which can be represented as $\mathcal{A} = \{a_1, a_2, \dots, a_m\}$, m is the number of local regions, and it is also the number of sampled points.

First, the point cloud $\mathcal{P}$ is processed using the Multilayer Perceptron (MLP) network with multiple layers of 1 × 1 convolutions to achieve the Spatial Features Transform Function (SFTF). For the spatial feature c_i of point p_i, the SFTF function can be expressed as: $v_i = tranform(c_i)$, where the function $tranform(x)$ transforms the d dimensional feature of each point into a higher-dimensional m feature, i.e., $\mathbb{R}^d \to \mathbb{R}^m$. The output $v_i \in \mathbb{R}^m$ of the transform function is a vector representing the correlation between point p_i and each of the m local regions. Extending the SFTF to the entire point cloud P, with the input as the features of n points $\mathcal{C} = \{c_1, c_2, \dots, c_n\}$, the corresponding $\mathcal{V}$ is obtained:

$$\mathcal{V} = transform(\mathcal{C}), \tag{5}$$

where the input $\mathcal{C} \in \mathbb{R}^{n \times d}$ and the output $\mathcal{V} \in \mathbb{R}^{n \times m}$ (the two-dimensional matrix with n rows).

Next, we use the $sigmoid$ function $q_i = \sigma(v_i)$ to obtain the probability vector q_i based on the correlation vector v_i. The probability vector q_i represents the probability that point p_i belongs to one of the m local regions $a_j (j = 1,2, \dots, m)$, where $q_i \in (0,1)$. When extended to the entire point cloud, the sigmoid function can be expressed as:

$$\mathcal{Q} = \sigma(\mathcal{V}), \tag{6}$$

where $\mathcal{Q} \in \mathbb{R}^{n \times m}$ is the probability matrix that represents the membership probabilities between each point in the point cloud $\mathcal{P}$ and each local region in the set of local regions $\mathcal{A}$. The columns of $\mathcal{Q}$ represent the probabilities of each point belonging to the corresponding local region, which can be denoted as $e_j \in \mathbb{R}^n$. Each value in e_j represents the probability of point p_i belonging to the local region a_j.

The column e_j in $\mathcal{Q}$ is sorted in descending order, and the indices of the top s probability values are selected. Here, the top s elements refer to the first s elements after sorting in descending order. The size of the point cloud for local region a_j is denoted as s. This process can be expressed as:

$$indices_j = argtop_s\left(desc\left(e_j\right)\right). \tag{7}$$

Here, $desc(x)$ is a function that sorts the elements of e_j in descending order, and $argtop_s$ is a function that returns the indices of the top s elements after sorting. $indices_j \in \mathbb{R}^s$.

Finally, the grouping and sampling of the point cloud $\mathcal{P}$ are completed as shown in the far right of Figure 6(b). The $indices_j$ are used to obtain the grouping indices for each sampled point, forming a grouping index matrix of size (n_{i+1}, k). Moreover, for each local region, the point p_i with the highest probability is selected as the downsampled point. This point is the one in the local region a_j that best matches the features of both the point and the local region. Here, $l = argtop_1\left(desc\left(e_j\right)\right)$ represents the index of the top-ranked point. Therefore, the set of downsampled points can be expressed as:

$$downsample = \left\{ p_l \,\middle|\, l = argtop_1\left(desc\left(e_j\right)\right) \right\} \; where \; j = 1,2, \dots, m. \tag{8}$$

The final output is the set of sampled points $\mathcal{P}_{i+1}(n_{i+1}, d_2)$, $\mathcal{P}_{i+1} \subset \mathcal{P}_i$.

Pre-LN consists of Layer Normalization [63] and a Linear layer, which helps reduce internal

covariate shift and improve the stability of the model during training. As shown in the upper part of Figure 6(a), we input the features $\mathcal{F}_i(n_i, d_1)$ into Layer Normalization to normalize and stabilize the distribution of data features for each sample point. Layer Normalization calculates the mean μ and standard deviation σ for all feature dimensions at each sample point.

$$\mu = \frac{1}{n_i}\sum_{j=1}^{n_i} f_j \,;\, \sigma = \sqrt{\frac{1}{n_i}\sum_{j=1}^{n_i}(f_j - \mu_i)^2}. \tag{9}$$

Here, f_j represents the feature of each point, and $f_j \in \mathcal{F}_i$. Then, through the Linear layer, the output dimensions remain unchanged at (n_i, d_1). To aggregate the projected features from the Pre-LN output while preserving feature invariance, we use the Max Pooling layer with grouping indices (n_{i+1}, k) generated by the grouping operation of PSNet. This aggregates the projected features and generates the output features $\mathcal{F}_{i+1}(n_{i+1}, d_1)$, where $\mathcal{F}_{i+1} \subset \mathcal{F}_i$.

2.3.3. The modified network

The network architecture utilizes the Stratified Transformer as the backbone and incorporates the Fast Downsample Layer to create the Plant Stratified Transformer for semantic segmentation of small-scale complex plant structures in point clouds. The overall network architecture is illustrated in Figure 7.

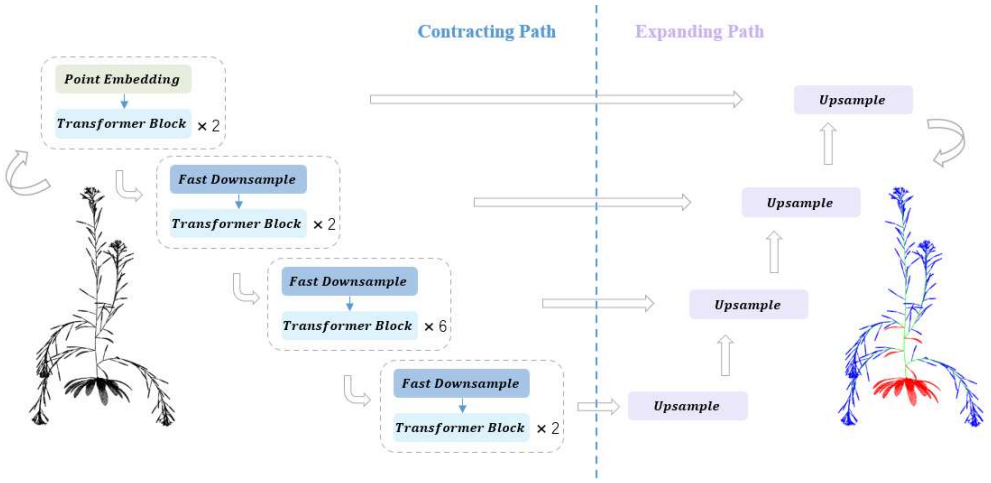

Figure 7. The overall structure of the Plant Stratified Transformer.

The network architecture resembles U-Net [64] and is divided into symmetric contracting and expanding paths. The left side of Figure 7 represents the contracting path, which is used to capture contextual information and perform hierarchical feature extraction, but it may lose some spatial information. At the beginning of the contracting path, the first-layer point embedding module aggregates the features of local neighbors for each point. Each subsequent module consists of a downsampling layer and several Transformer modules that capture local and long-range dependencies in the point cloud. The right side of Figure 7 represents the expanding path, which is used to upsample the features extracted from the contracting path and achieve precise localization of the segmented parts in the point cloud. The expanding path comprises multiple Upsample Layers, which densify features layer by layer. However, upsampling alone cannot recover spatial information. Hence, skip connections are employed to output shallow features from each stage of the contracting path to the

corresponding upsampling layer in the expanding path. The network preserves more high-resolution details from shallow feature maps by fusing shallow and deep features in the Upsample Layers, thereby enhancing the accuracy of semantic segmentation.

The first-layer point embedding module in the network aggregates local features from adjacent points in the point cloud to enhance the model's generalization and expressive capabilities. The Transformer blocks employ Stratified Self-attention and Stratified Key-sampling Strategy [52] to increase the effective receptive field and enable the effective aggregation of long-range contextual information by query features.

2.4. Evaluation metrics

In order to verify the semantic segmentation performance of the Plant Stratified Transformer(our) on plant point clouds, we used four metrics: Precision (Prec), Recall (Rec), F1, and Intersection over Union (IoU) to compare the success rate of organ segmentation in plants. For all four semantic metrics (expressed as percentages), a higher value indicates better performance. Specifically, Prec represents the ratio of correctly classified points in a semantic class to all points predicted by the network. Rec reflects the ratio of correctly classified points in this semantic class to the total number of points in this class according to the true labels. F1 is a comprehensive metric calculated as the harmonic mean of Prec and Rec. For each semantic class, IoU reflects the degree of overlap between the predicted region of each semantic class and the corresponding true region. The four metrics are defined as follows:

$$Prec_c = \frac{TP_c}{TP_c+FN_c}, \tag{10}$$

$$Rec_c = \frac{TP_c}{TP_c+FP_c}, \tag{11}$$

$$F1_c = 2 \cdot \frac{Pre\ _c \times Rec_c}{Prec_c+Rec_c}, \tag{12}$$

$$IoU_c = \frac{TP_c}{TP_c+FN_c+FP_c}, \tag{13}$$

where TP_c represents the true positive point count of the current semantic class, FP_c represents the false positive point count of the current class, and FN_c represents the false negative point count. c is the semantic label, $c \in \{leaf, stem, flower\}$.

3. Results

3.1. Data preparation and training environment

In this study, we used a generated virtual Arabidopsis dataset and a real Arabidopsis dataset with semantic labels. The latter was a dataset we created from 10 real 3D Arabidopsis plants selected from Ziamtsov et al. [65] dataset.

Virtual Arabidopsis dataset: We simulated Arabidopsis models with a developmental period of approximately 6 weeks using the L-Py framework. The generated models were converted into fully annotated point clouds using the surface-weighted sampling method. To simplify the labeling

categories of the models, we unified the siliques and flowers of Arabidopsis as 'flower'. The final generated Arabidopsis models had semantic labels for 'leaf', 'stem', and 'flower'. The downsampling method described earlier was applied to reduce the number of points in the generated point cloud models. Through these steps, a total of 930 Arabidopsis point cloud models with diverse morphologies were generated. Figure 8(b) displays different morphological Arabidopsis point cloud models, each containing 8192 points. The information of an individual point included coordinates and semantic labels. The semantic labels 'leaf', 'stem', and 'flower' were represented by red, green, and blue points in the point cloud, respectively, and were denoted as '0', '1' and '2' in the point information, as shown in Figure 8(a). Since we could directly generate point cloud models with significantly different morphological structures, it reduced the possibility of overfitting and eliminates the need for traditional data augmentation operations. We divided the dataset into a training set and a validation set in a ratio of 4:1.

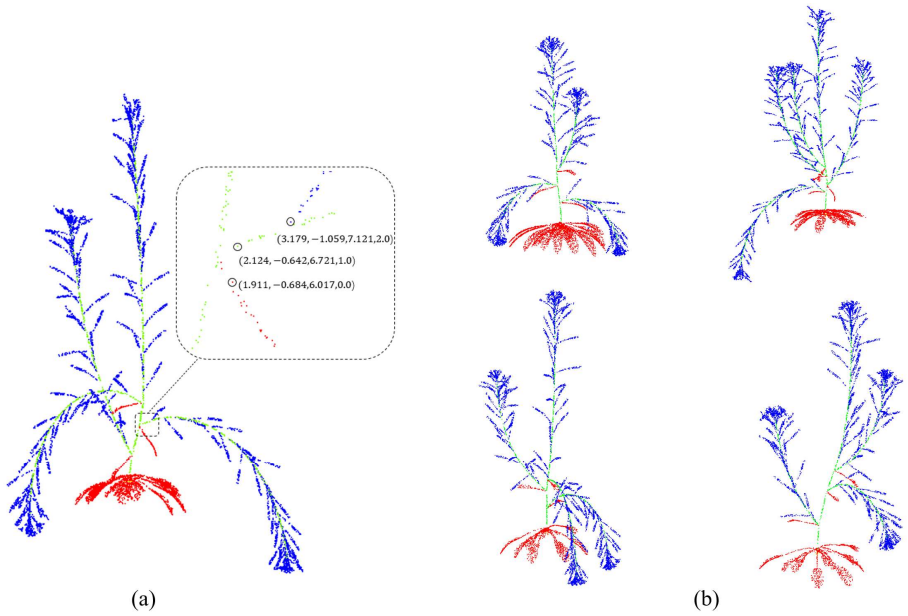

Figure 8. Fully annotated Arabidopsis point cloud model. (a) represents the information representation of points in the point cloud. (b) shows Arabidopsis point clouds with different structures.

Real Arabidopsis dataset: The dataset from Ziamtsov et al. [65] comprises high-resolution measurement data of plant structures generated using 3D scanning techniques. We selected 47 Arabidopsis 3D scanning models from this dataset with distinct structural features, as shown in Figure 9(a). Compared to our generated virtual Arabidopsis, these real Arabidopsis models exhibited more complex morphological structures. The proportion of siliques and flowers was lower, while the rosette part had a higher proportion.

We used CloudCompare to convert these models into point clouds and performed semantic annotation on leaves, stems, and flowers. For data augmentation, random rotations, cropping, and stitching operations were applied to point clouds to enhance the diversity among the data. Subsequently, we downsampled the point cloud to 8192 points using the method in Section 2.2, as shown in Figure 9(b). To ensure the separation of training and test dataset, 27 point clouds were randomly selected from the 47

real point clouds and augmented five times, resulting in 135 point clouds for the training set. The remaining 20 real point clouds were used as the test set.

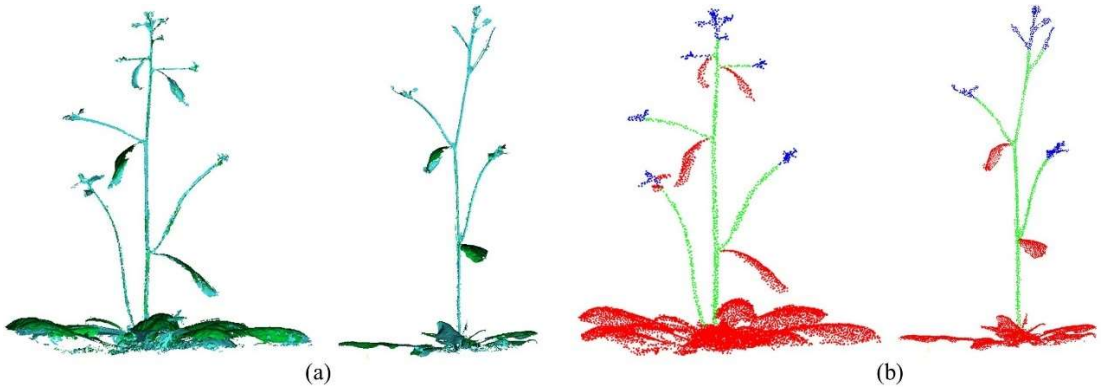

Figure 9. 3D scanning models and point clouds of real Arabidopsis.

Due to the scarcity of real Arabidopsis, overfitting was likely to occur during network training, leading to an inability to achieve the expected performance. We merged the training sets of virtual and real Arabidopsis to create a mixed training set, comprising 744 virtual point clouds and 135 real point clouds. The validation set consists of the validation set of virtual Arabidopsis, comprising 186 virtual point clouds. The test set consists of the test set of real Arabidopsis, comprising 20 real point clouds.

Properly setting the learning rate is crucial during the network training phase. A high learning rate could lead to gradient explosions, causing significant oscillation in the loss and hindering model convergence. Conversely, A low learning rate slows down the model's learning speed and increases training time. For improved network convergence, we applied a learning rate decay with a fixed step size of 0.5 every 20 epochs, starting from an initial learning rate of 0.001. We trained the network with a batch size of 8 for a total of 250 epochs in the training environment specified in Table 1. For testing, we conducted training and evaluation on the Arabidopsis dataset using PointNet++ [38], PAConv [66] and the Stratified Transformer with identical parameters and environment, and then compared their performance with our improved network.

Table 1. Training environment.

Name	Parameter
CPU	Intel(R) Core(TM) i7-7700 CPU
GPU	GeForce GTX 1070
Memory	16 GB
Operating system	Ubuntu 20.04
Deep learning framework	Pytorch 1.8.0
Programming language	Python 3.9

3.2. Validation on virtual data

In this section, we individually trained four different networks on the mixed training set and validated them using the virtual Arabidopsis validation set. We compared the segmentation

performance of our network with the other three networks. Table 2 presents the segmentation results of the improved network, Plant Stratified Transformer (ours), and three other segmentation methods for comparison. The table displayed the values of various metrics for the epoch with the highest IoU after network convergence. The values related to 'stem' are significantly lower compared to 'leaf' and 'flower' in the table. This is due to the stem category having fewer points in each Arabidopsis point cloud compared to the other two categories, leading to potential confusion during network training. Furthermore, our network achieve performance similar to the original network in mean precision, mean recall, mean F1-score, and mean IoU, with values of 88.12, 91.44, 89.64 and 85.56%. The differences between the two networks in these metrics are only 0.28, 0.22, 0.08 and 0.09%, indicating that the inclusion of the Fast Downsample Layer do not affect the performance of the network in semantic segmentation. Both networks outperform PointNet++ and PAConv, as the Transformer module effectively capture long-range contextual information, resulting in a larger receptive field and improved generalization ability for the model. In contrast, PointNet++ and PAConv rely on local feature aggregation and do not directly establish long-range dependencies, making it difficult to capture long-range contextual information. Consequently, due to the lower number of points in the stem category, PointNet++ and PAConv exhibit poorer performance in the metrics related to the stem category compared to the other two networks, whereas their performance differences in the other categories are less significant. PAConv, which is an improvement over the original backbone network PointNet, performs slightly better than PointNet++ but do not fully utilize global information. From the table, it could be seen that PointNet++ performs the worst, especially in the stem category, indicating that this network is not suitable for handling sparse plant point clouds with complex structures.

Table 2. The comparison of semantic segmentation across the four networks on the validation set. The best results are in boldface.

		PointNet++	PAConv	Stratified Transformer	Plant Stratified Transformer (ours)
Prec (%)	Leaf	95.78	95.39	98.57	98.49
	Stem	69.48	73.18	77.59	78.18
	Flower	91.26	94.23	96.32	96.79
	Mean	85.51	87.60	90.83	91.15
Rec (%)	Leaf	94.67	95.48	97.83	97.93
	Stem	72.35	75.83	80.41	80.37
	Flower	92.94	95.21	98.15	98.61
	Mean	86.65	88.84	92.13	92.30
F1 (%)	Leaf	95.22	95.43	98.20	98.21
	Stem	70.89	74.48	78.97	79.26
	Flower	92.09	94.72	97.23	97.69
	Mean	83.05	84.96	88.59	91.72
IoU (%)	Leaf	90.88	91.27	96.46	96.48
	Stem	54.90	59.34	65.25	65.65
	Flower	85.34	89.97	94.60	95.49
	Mean	77.04	80.19	85.44	85.87

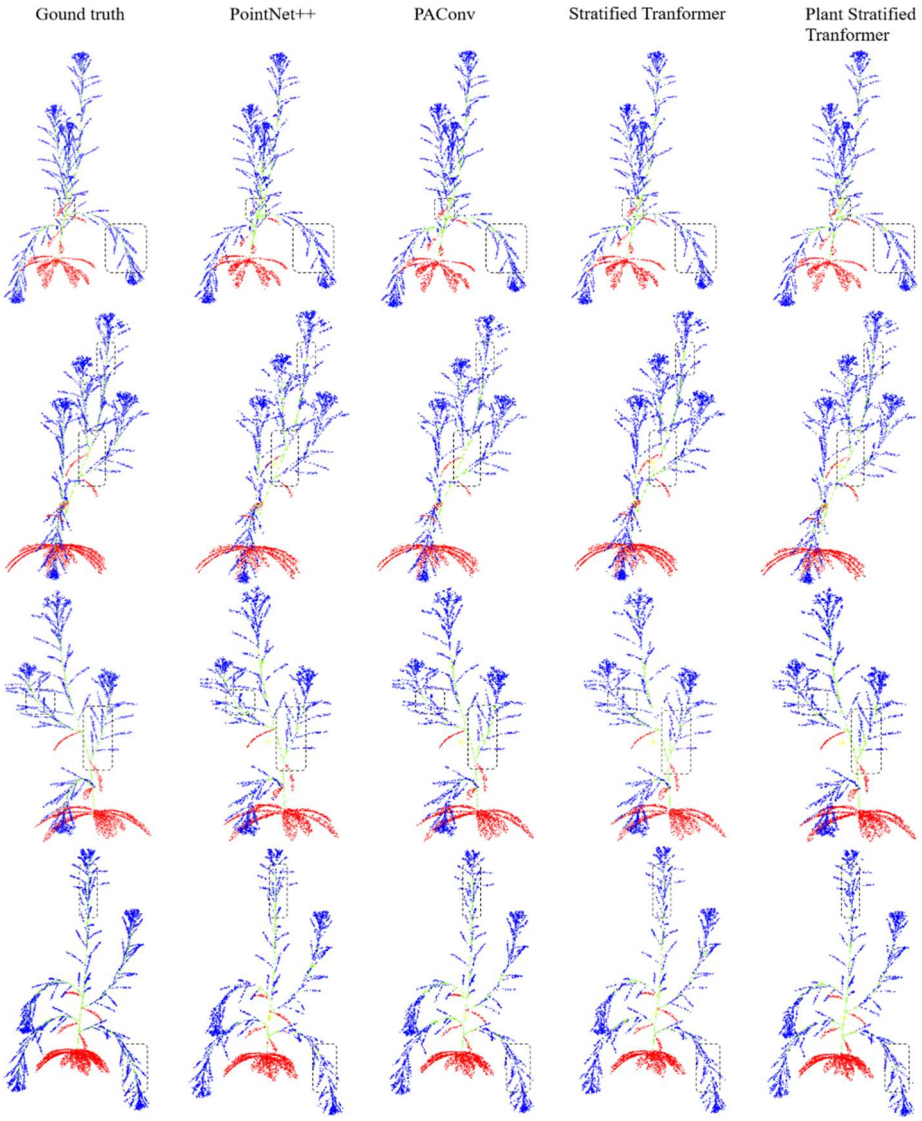

Figure 10. The qualitative segmentation comparison of the four networks on the validation set.

The visualization results of semantic segmentation for the four networks are shown in Figure 10. The leftmost images represent the ground truth, and the images on the right display the test results of different networks. We select four Arabidopsis point clouds with significant morphological differences to demonstrate the visualization results of semantic segmentation. From Figure 10, it is visually evident that the number of points belonging to the stem is significantly fewer than those belonging to the leaf and flower. The segmentation performance is clearly the worst at the boundaries between different organ structures, leading to confusion, especially at the boundaries between the stem and the other two structures. The figure also shows that the segmentation of the leaf was superior to that of the stem and flower. This is because the leaf had fewer intersections with the stem and flower, it has a clear structure, and there are also more points.

3.3. Segmentation of real data

We tested the networks trained on the mixed dataset on the real Arabidopsis testing set, and the

segmentation results are shown in Table 3, compared with the other three networks. Clearly, the segmentation performance of all networks significantly decreases. PointNet++, PAConv and Stratified Transformer had mean IoU of 57.93, 61.02 and 69.86%, respectively, which were decreased by 19.11, 19.17 and 15.58% compared to the segmentation results on virtual Arabidopsis. The segmentation performance of PointNet++ and PAConv was notably poor, failing to meet the requirements for semantic segmentation, possibly due to limitations in their generalization abilities. The improved network achieved mean Prec, Rec, F1-score and IoU of 80.42, 82.18, 81.29 and 70.04%, respectively, which decreased by 10.73, 10.12, 10.43 and 15.83% compared to the segmentation results on virtual Arabidopsis. Both the Stratified Transformer and our network's mean IoU could maintain around 70%, meeting the requirements for semantic segmentation and showcasing good generalization capability. Furthermore, Table 3 shows that the primary reason for the decrease in Mean IoU is a significant drop in the metrics related to the 'Flower' category. This is due to the scarcity of the 'flower' label in the tested real Arabidopsis compared to the virtual Arabidopsis.

Table 3. The comparison of semantic segmentation across the four networks on the test set. The best results are in boldface.

		PointNet++	PAConv	Stratified Transformer	Plant Stratified Transformer (ours)
Prec (%)	Leaf	92.54	94.42	97.51	97.93
	Stem	72.58	75.34	80.21	79.85
	Flower	65.35	69.10	75.12	74.81
	Mean	76.82	79.62	84.28	84.20
Rec (%)	Leaf	89.25	93.25	95.82	96.14
	Stem	66.89	66.48	80.22	79.68
	Flower	60.57	67.31	72.93	73.28
	Mean	72.24	75.68	82.99	83.03
F1 (%)	Leaf	90.87	93.83	96.66	97.03
	Stem	69.62	70.63	80.21	79.76
	Flower	62.87	68.19	74.01	74.04
	Mean	74.45	77.55	83.63	83.61
IoU (%)	Leaf	83.26	88.38	93.53	94.23
	Stem	53.40	54.60	66.97	66.34
	Flower	45.85	51.74	58.74	58.78
	Mean	60.83	64.91	73.08	73.11

Figure 11 presents a qualitative comparison of the performance of four networks on the real Arabidopsis testing set. It can be visually observed that the segmentation is poorest at boundaries between different organs. Compared to virtual Arabidopsis, the testing performance on real Arabidopsis is relatively poor due to morphological differences between the real and virtual Arabidopsis used in the experiments. This difference primarily lies in the virtual Arabidopsis having far more flowers and siliques than the real Arabidopsis, with significantly fewer points on the rosette's leaf compared to the real Arabidopsis. Similarly, there are more points on the rosette's leaf of the real Arabidopsis. This results in a smaller proportion of non-rosette points in the real Arabidopsis under the condition of the same point cloud size. Consequently, fewer non-rosette point features are captured by

the network compared to virtual Arabidopsis, leading to a further decline in segmentation performance.

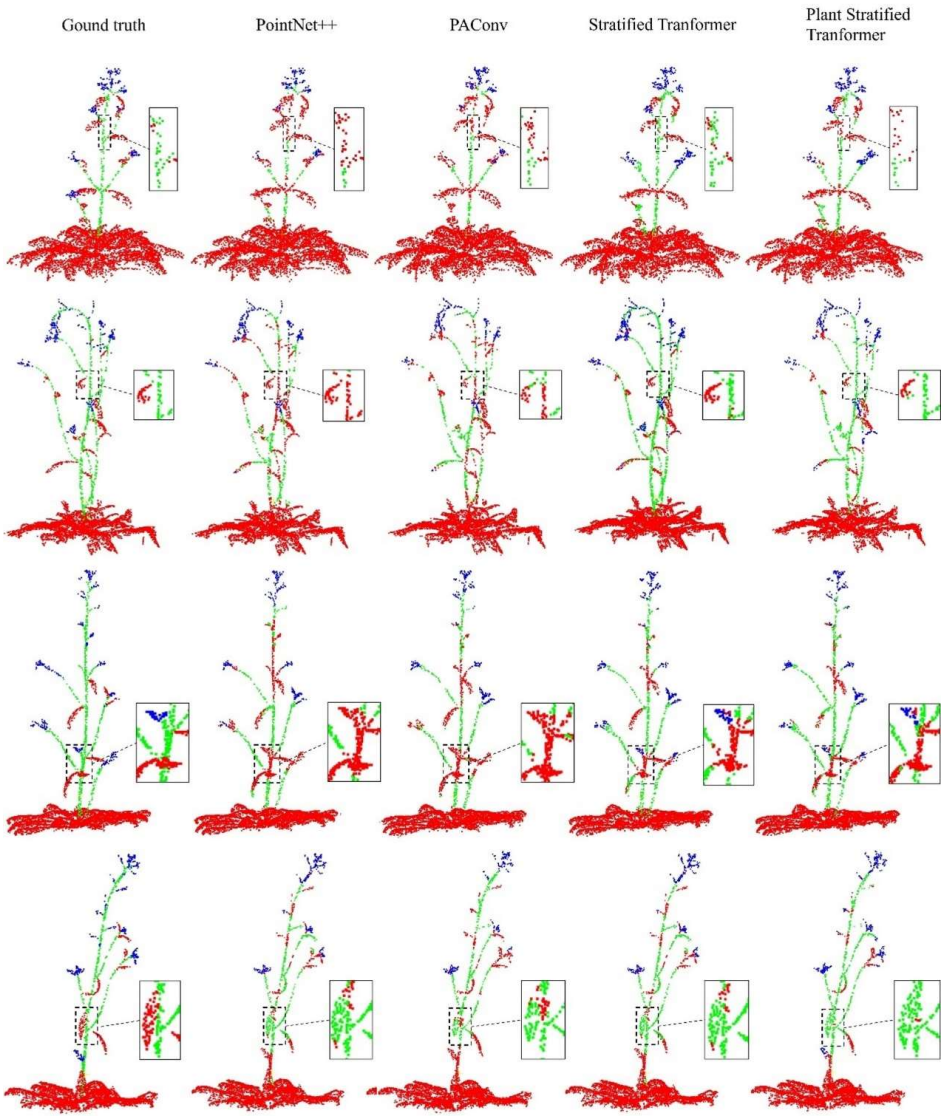

Figure 11. The qualitative comparison of segmentation on the test set by the four networks.

3.4. Time-consuming analysis

To assess the efficiency of our improved network on our dataset, we recorded the Training time and Inference time for each network during the experiments. Training time referred to the time taken during the training phase, from extracting a batch of data from the training set to feeding it into the neural network and obtaining the output results of the model during each iteration. Inference time was the time taken for forward propagation, which was the duration it takes for the model to process new input data and produce output predictions during the testing phase. From Table 4, it can be observed that, using the FPS and kNN sampling and grouping method, both the Training time and Inference time for Stratified Transformer are longer compared to PointNet++ and PAConv. This was because the Transformer block required processing a large number of point-wise self-attention weights.

Table 4. The mean training time (ms) and inference time (ms) for each method. The best results are in boldface.

Methods	Training time	Inference time
PointNet++	792.7	761.5
PAConv	678.8	652.3
Stratified Tranformer	1035.2	869.7
Plant Stratified Tranformer(our)	714.3	597.9

The training time and inference time of our improved network were 714 and 597.9 ms, respectively, which were reduced by 320.9 and 271.8 ms compared to the original network. PAConv had the shortest training time, 678.8 ms, which was 35.5 ms less than our network. However, our network's inference time was 54.4 ms less than PAConv, and our inference time was the optimal among the compared networks. All of these demonstrated that, compared to the original network, the Fast Downsample Layer replacing the original downsampling layer had shortened the time required for point cloud sampling and grouping.

4. Discussion

4.1. Ablation study

In this section, we designed several independent ablation experiments to verify the effectiveness of the Fast Downsample Layer proposed in the Plant Stratified Transformer. This includes assessing the effectiveness of sampling and grouping in PSNet, as well as the functionality of the Pre-LN module. For the sake of comparison and demonstration, we used traditional sampling and grouping methods, FPS and KNN, as benchmarks, and named the sampling and grouping parts of PSNet as S1 and G1, FPS as S2, and kNN as G2. Since G1 needs to work in conjunction with S1, it cannot exist independently of S1. Ablation experiments for semantic segmentation are presented in Table 5, and ablation experiments for mean training and inference time in semantic segmentation are shown in Table 6.

In Tables 5 and 6, the "Ver" column provides the names of the ablated network versions. We compared eight version networks named "V1" to "V5" with the complete network ("V0"). Each version was created by removing or replacing existing modules from the original Plant Stratified Transformer. In the comparison of different sampling and grouping methods, we found that sampling and grouping methods can slightly improve the effectiveness of the network model. Removing or adding the Pre-LN module also had an impact on the network's performance, possibly because normalized features are more conducive to the stability of the model during training, to some extent improving the model's effectiveness. This may be because the Fast Downsample Layer can provide more suitable local grouping compared to the original model.

In Table 6, we compared the impact of different modules on the training and inference time. Pre-LN somewhat accelerated the network's speed because normalization can expedite network convergence. Most importantly, our sampling and grouping methods are significantly faster than the combination of FPS and kNN. This is because the time complexity of FPS + kNN is much greater than that of PSNet. The time complexity of FPS is $O(n^2)$ [61]. The time complexity of kNN includes $O(nm)$ for distance calculation and $O(n\log_2 n)$ for heap sorting. Therefore, the time complexity of FPS+kNN is $O(n^2 + nm + n\log_2 n)$. In contrast, the time complexity of PSNet is $O(nm + n\log_2 n)$, which includes

SFTF and heap sorting. (n represents the number of sampled points, and m represents the number of groups).

Table 5. The ablation analysis of our network on the validation set. The best results are in boldface.

	Ver	Sampling	Grouping	Pre-LN	Leaf	Stem	Flower	Mean
Prec	V0	S1	G1	√	98.49	78.18	96.79	91.15
(%)	V1	S1	G1		98.03	77.80	96.23	90.69
	V2	S1	G2	√	98.35	77.93	96.64	90.97
	V3	S1	G2		98.12	77.67	96.19	90.66
	V4	S2	G2	√	98.57	77.59	96.32	90.83
	V5	S2	G2		97.93	77.14	96.12	90.40
Rec	V0	S1	G1	√	97.93	80.37	98.61	92.30
(%)	V1	S1	G1		97.34	79.87	98.13	91.78
	V2	S1	G2	√	97.91	80.29	98.28	92.16
	V3	S1	G2		97.30	80.03	98.06	91.80
	V4	S2	G2	√	97.83	80.41	98.15	92.13
	V5	S2	G2		97.42	79.81	98.01	91.75
F1 (%)	V0	S1	G1	√	98.21	79.26	97.69	91.72
	V1	S1	G1		97.68	78.82	97.17	91.23
	V2	S1	G2	√	98.13	79.09	97.45	91.56
	V3	S1	G2		97.71	78.83	97.12	91.22
	V4	S2	G2	√	98.20	78.97	97.23	91.47
	V5	S2	G2		97.67	78.45	97.06	91.06
IoU	V0	S1	G1	√	96.48	65.65	95.49	85.87
(%)	V1	S1	G1		95.47	65.05	94.50	85.01
	V2	S1	G2	√	96.33	65.42	95.03	85.59
	V3	S1	G2		95.52	65.06	94.39	84.99
	V4	S2	G2	√	96.46	65.25	94.60	85.44
	V5	S2	G2		95.45	64.54	94.28	84.76

Table 6. The ablation analysis of our network on the training time (ms) and inference time (ms). The best results are in boldface.

Ver	Sampling	Grouping	Pre-LN	Training time	Inference time
V0	S1	G1	√	721.4	613.9
V1	S1	G1		735.9	624.5
V2	S1	G2	√	754.8	643.5
V3	S1	G2		773.5	659.4
V4	S2	G2	√	1041.5	873.2
V5	S2	G2		1059.8	894.3

4.2. The effect of virtual data on network training

To study the impact of the ratio of virtual to real data on network training and validate whether

the inclusion of virtual plants in the mixed dataset enhances accuracy, we divided the training set into V, R1, R2 and R3, as shown in Table 7. R1, R2 and R3 were obtained by augmenting data from 9, 18 and 27 real plants, respectively, five times. These four parts were combined to form 7 training sets, each trained separately. The seven sets of data included virtual data (V), real data (R1, R2, R3), and mixed data (V + R1, V + R2, V + R3). The test set remained as 20 real point clouds. The four networks were trained separately seven times, and the variations in testing results were compared when using different combinations of training sets.

Table 7. Partitioning of training data.

	V	R1	R2	R3
Data	Virtual	Real	Real	Real
Training	744	9:45	18:90	27:135

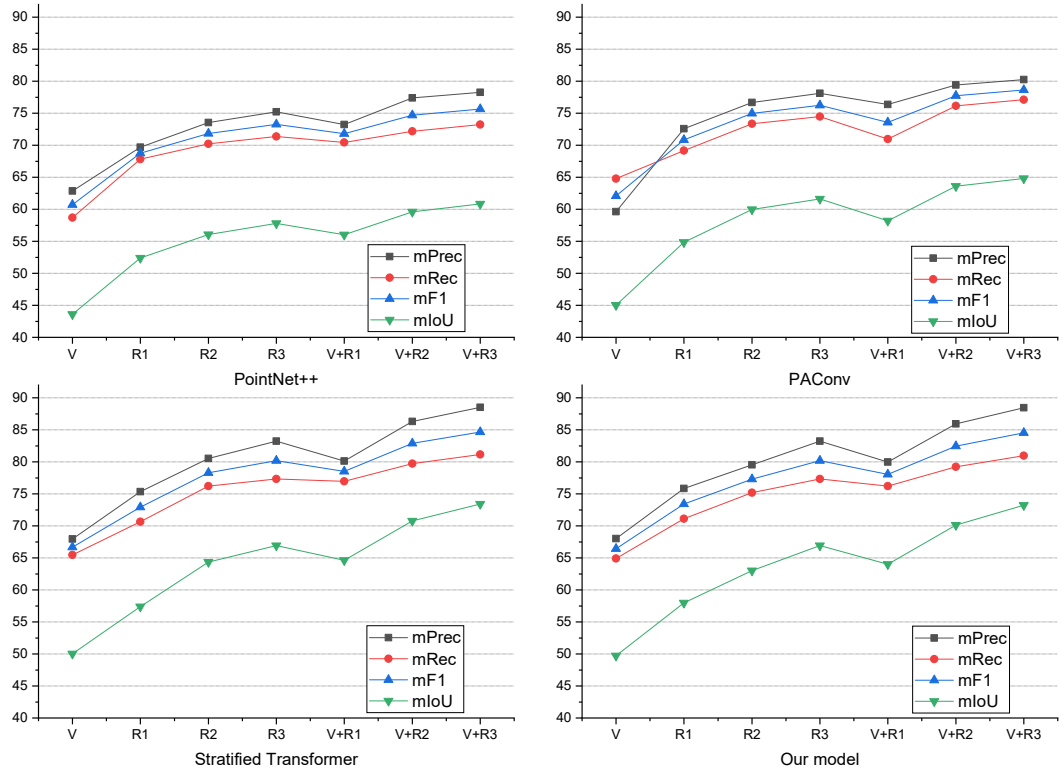

Figure 12. The variation in test results of the four networks on different combinations of training sets.

Figure 12 presents the test results of models trained by various networks on different combinations of training sets. PointNet++ shows the smallest increase in various metrics as the combinations change. In contrast, Stratified Transformer and our model exhibit the largest increases. This indicates that the latter two can learn more features similar to both virtual and real data. Training exclusively with virtual data (V) yields unsatisfactory results, especially as PointNet++ fails to achieve even a 45% mIoU. Training exclusively with different quantities of real data (R1, R2 and R3) shows improved performance with an increase in the amount of real data. The metrics between R1 and R2 clearly show a higher improvement compared to that between R2 and R3. This suggests that the

improvement in network training effectiveness diminishes with the increase in real data. In the case of the combination of R1, R2, R3 and V, the training performance is evidently better than training with real data alone. This indicates that in this study, virtual plant data can indeed enhance the overall training effectiveness when combined with a certain amount of real data. The combination of a large amount of virtual data with a small amount of real data achieves acceptable training results.

4.3. Limitations

Our virtual plants have certain shortcomings when compared to other virtual plants. For instance, Morel et al. [67] used a TLS simulator to generate point clouds from virtual tree mesh models. This method simulates LiDAR scanning of trees at certain heights and distances. The references [68,69] considered the impact of simulating the distance between real scanning devices and plant objects on point cloud density. However, our method does not simulate the position of scanning devices on point clouds. In future work, we can introduce variable point density and artificial noise into point clouds by simulating acquisition systems such as ToF cameras and LiDAR in a virtual environment.

Point clouds obtained from scanning real plants are typically very large, whereas the maximum point cloud size our network takes as input is 8912. For plants with dense foliage or larger leaves, this can result in very sparse leaf point clouds, thus affecting the final segmentation results. The structure of real Arabidopsis is more complex. In real environments, it is challenging to fully capture the stems, leading to significant differences in the proportions of flowers and stems in various Arabidopsis. Typically, Arabidopsis rosette leaves are abundant, causing the proportion of non-rosette features to be too low in a fixed-size point cloud. This results in the loss of more phenotypic features. In future work, Arabidopsis, such as rosette plants, we can separate the rosette and non-rosette parts before segmenting individually.

5. Conclusions

In this study, virtual Arabidopsis models generated by L-system were used as experimental data. This allowed for the involvement of virtual plants in network training alongside real plants, addressing the scarcity of Arabidopsis point cloud data. Since the generated virtual plants came with semantic labels, this eliminated the need for manual annotation. We proposed the Plant Stratified Transformer for semantic segmentation tasks on plants. Our network was based on the Stratified Transformer backbone, incorporating the Fast Downsample Layer to accelerate network speed. In our dataset, the training set was a mixture of virtual and real Arabidopsis point clouds, while the test set consisted of real data. The improved network, alongside PointNet++, PAConv and the original Stratified Transformer, was trained and tested on this dataset. In terms of segmentation performance, the improved network was comparable to the original network but significantly outperforms the other two. Regarding time consumption, the improved network had much shorter training and inference times compared to the original network. This indicated that the improved network enhanced training and inference efficiency while maintaining the segmentation performance of the original network. The research results thoroughly demonstrated the significant potential of virtual plants and deep learning methods in rapidly extracting plant phenotypes. Synthetic virtual data could effectively facilitate the training of deep learning models, thereby reducing costs and time. This not only has a positive impact on plant phenotype research but also injects new vitality into the field's development.

In future work, we plan to acquire 3D point cloud models by scanning Arabidopsis to obtain additional

real data for network training. We will improve or design new deep learning architectures to handle plant point cloud models more efficiently, meeting the real-time requirements of practical applications.

Use of AI tools declaration

The authors declare that they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by National Key Research and Development Program of China (Project no. 2023YFD200140301), Project of China Construction Center Construction Engineering Co., LTD (Project no. ZX&AZ02202200016001), Shandong Province Postgraduate Quality Case Library Project (Project no. SDYAL2022143), National Key R&D Program of China (Project no. 2022YFE0125800), Basic Scientific Research Project of Liaoning Provincial Department of Education (Project no. LJKQZ20222458) and the National Modern Agricultural Industry Technology System Post Scientist Project (CARS-13-National Peanut Industry Technology System-Sowing and Field Management Mechanization Post).

Conflict of interest

The authors declare that there are no conflicts of interest.

References

1. R. Pieruschka, U. Schurr, Plant phenotyping: past, present, and future, *Plant Phenomics*, **2019** (2019). https://doi.org/10.34133/2019/7507131
2. C. Costa, U. Schurr, F. Loreto, P. Menesatti, S. Carpentier, Plant phenotyping research trends, a science mapping approach, *Front. Plant Sci.*, **9** (2019), 1933. https://doi.org/10.3389/fpls.2018.01933
3. A. K. Singh, B. Ganapathysubramanian, S. Sarkar, A. Singh, Deep learning for plant stress phenotyping: trends and future perspectives, *Trends Plant Sci.*, **23** (2018), 883–898. https://doi.org/10.1016/j.tplants.2018.07.004
4. S. Arya, K. S. Sandhu, J. Singh, S. Kumar, Deep learning: as the new frontier in high-throughput plant phenotyping, *Euphytica*, **218** (2022), 47. https://doi.org/10.1007/s10681-022-02992-3
5. S. Bhagat, M. Kokare, V. Haswani, P. Hambarde, R. Kamble, Eff-UNet++: A novel architecture for plant leaf segmentation and counting, *Ecol. Inf.*, **68** (2022), 101583. https://doi.org/10.1016/j.ecoinf.2022.101583
6. K. Khan, R. U. Khan, W. Albattah, A. M. Qamar, End-to-end semantic leaf segmentation framework for plants disease classification, *Complexity*, **2022** (2022). https://doi.org/10.1155/2022/1168700
7. D. Zendler, N. Malagol, A. Schwandner, R. Töpfer, L. Hausmann, E. Zyprian, High-throughput phenotyping of leaf discs infected with grapevine downy mildew using shallow convolutional neural networks, *Agronomy*, **11** (2021), 1768. https://doi.org/10.3390/agronomy11091768

8. J. Wu, C. Wen, H. Chen, Z. Ma, T. Zhang, H. Su, et al., DS-DETR: A model for tomato leaf disease segmentation and damage evaluation, *Agronomy*, **12** (2022), 2023. https://doi.org/10.3390/agronomy12092023
9. Y. Wu, L. Xu, Crop organ segmentation and disease identification based on weakly supervised deep neural network, *Agronomy*, **9** (2019), 737. https://doi.org/10.3390/agronomy9110737
10. Z. Li, R. Guo, M. Li, Y. Chen, G. Li, A review of computer vision technologies for plant phenotyping, *Comput. Electron. Agric.*, **176** (2020), 105672. https://doi.org/10.1016/j.compag.2020.105672
11. Y. Jiang, C. Li, Convolutional neural networks for image-based high-throughput plant phenotyping: a review, *Plant Phenomics*, **2020** (2020). https://doi.org/10.34133/2020/4152816
12. W. D. Kissling, Y. Shi, Z. Koma, C. Meijer, O. Ku, F. Nattino, et al., Laserfarm–A high-throughput workflow for generating geospatial data products of ecosystem structure from airborne laser scanning point clouds, *Ecol. Inf.*, **72** (2022), 101836. https://doi.org/10.1016/j.ecoinf.2022.101836
13. J. Zhou, X. Fu, S. Zhou, J. Zhou, H. Ye, H. T. Nguyen, Automated segmentation of soybean plants from 3D point cloud using machine learning, *Comput. Electron. Agric.*, **162** (2019), 143–153. https://doi.org/10.1016/j.compag.2019.04.014
14. X. Ma, K. Zhu, H. Guan, J. Feng, S. Yu, G. Liu, Calculation method for phenotypic traits based on the 3D reconstruction of maize canopies, *Sensors*, **19** (2019), 1201. https://doi.org/10.3390/s19051201
15. S. Wu, W. Wen, Y. Wang, J. Fan, C. Wang, W. Gou, et al., MVS-Pheno: a portable and low-cost phenotyping platform for maize shoots using multiview stereo 3D reconstruction, *Plant Phenomics*, **2020** (2020). https://doi.org/10.34133/2020/1848437
16. H. You, Y. Liu, P. Lei, Z. Qin, Q. You, Segmentation of individual mangrove trees using UAV-based LiDAR data, *Ecol. Inf.*, (2023), 102200. https://doi.org/10.1016/j.ecoinf.2023.102200
17. P. Li, X. Zhang, W. Wang, H. Zheng, X. Yao, Y. Tian, et al., Estimating aboveground and organ biomass of plant canopies across the entire season of rice growth with terrestrial laser scanning, *Int. J. Appl. Earth Obs. Geoinf.*, **91** (2020), 102132. https://doi.org/10.1016/j.jag.2020.102132
18. Y. Sun, Y. Luo, Q. Zhang, L. Xu, L. Wang, P. Zhang, Estimation of crop height distribution for mature rice based on a moving surface and 3D point cloud elevation, *Agronomy*, **12** (2022), 836. https://doi.org/10.3390/agronomy12040836
19. F. Tardieu, Virtual plants: modelling as a tool for the genomics of tolerance to water deficit, *Trends Plant Sci.*, **8** (2003), 9–14. https://doi.org/10.1016/S1360-1385(02)00008-0
20. P. Prusinkiewicz, Graphical applications of L-systems, in *Proceedings of Graphics Interface*, Canadian Information Processing Society, Vancouver, Canada, **86** (1986), 247–253.
21. R. Karwowski, P. Prusinkiewicz, Design and implementation of the L+ C modeling language, *Electron. Notes Theor. Comput. Sci.*, **86** (2003), 134–152. https://doi.org/10.1016/S1571-0661(04)80680-7
22. F. Boudon, C. Pradal, T. Cokelaer, P. Prusinkiewicz, C. Godin, L-Py: an L-system simulation framework for modeling plant architecture development based on a dynamic language, *Front. Plant Sci.*, **3** (2012), 76. https://doi.org/10.3389/fpls.2012.00076
23. R. Barth, J. IJsselmuiden, J. Hemming, E. J. V. Henten, Synthetic bootstrapping of convolutional neural networks for semantic plant part segmentation, *Comput. Electron. Agric.*, **161** (2019), 291–304. https://doi.org/10.1016/j.compag.2017.11.040

24. M. Cieslak, N. Khan, P. Ferraro, R. Soolanayakanahally, S. J. Robinson, I. Parkin, et al., L-system models for image-based phenomics: case studies of maize and canola, *In Silico Plants*, **4** (2021), diab039. https://doi.org/10.1093/insilicoplants/diab039
25. E. Fiestas, O. E. Ramos, S. Prado, RPA and L-system based synthetic data generator for cost-efficient deep learning model training, in *2021 IEEE 3rd Eurasia Conference on IOT, Communication and Engineering (ECICE)*, National Formosa University, Yunlin, Taiwan, (2021), 645–650. https://doi.org/10.1109/ECICE52819.2021.9645719
26. D. Ward, P. Moghadam, N. Hudson, Deep leaf segmentation using synthetic data, preprint, arXiv: 1807.10931. https://doi.org/10.48550/arXiv.1807.10931
27. R. Barth, J. IJsselmuiden, J. Hemming, E. J. V. Henten, Data synthesis methods for semantic segmentation in agriculture: A Capsicum annuum dataset, *Comput. Electron. Agric.*, **144** (2018), 284–296. https://doi.org/10.1016/j.compag.2017.12.001
28. J. Ubbens, M. Cieslak, P. Prusinkiewicz, I. Stavness, The use of plant models in deep learning: an application to leaf counting in rosette plants, *Plant Methods*, **14** (2018), 1–10. https://doi.org/10.1186/s13007-018-0273-z
29. K. Turgut, H. Dutagaci, D. Rousseau, RoseSegNet: An attention-based deep learning architecture for organ segmentation of plants, *Biosyst. Eng.*, **221** (2022), 138–153. https://doi.org/10.1016/j.biosystemseng.2022.06.016
30. A. Chaudhury, P. Hanappe, R. Azaïs, C. Godin, D. Colliaux, Transferring PointNet++ segmentation from virtual to real plants, in *ICCV 2021-International Conference on Computer Vision*, IEEE computer society, Montreal, (2021), 13.
31. Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, M. Bennamoun, Deep learning for 3d point clouds: A survey, *IEEE Trans. Pattern Anal. Mach. Intell.*, **43** (2020), 4338–4364. https://doi.org/10.1109/TPAMI.2020.3005434
32. H. Su, S. Maji, E. Kalogerakis, E. Learned-Miller, Multi-view convolutional neural networks for 3d shape recognition, in *Proceedings of the IEEE International Conference on Computer Vision*, IEEE computer society, Montreal, QC, Canada, (2015), 945–953. https://doi.org/10.1109/ICCV.2015.114
33. W. Shi, R. van de Zedde, H. Jiang, G. Kootstra, Plant-part segmentation using deep learning and multi-view vision, *Biosyst. Eng.*, **187** (2019), 81–95. https://doi.org/10.1016/j.biosystemseng.2019.08.014
34. X. Wang, C. Wang, B. Liu, X. Zhou, L. Zhang, J. Zheng, et al., Multi-view stereo in the deep learning era: A comprehensive review, *Displays*, **70** (2021), 102102. https://doi.org/10.1016/j.displa.2021.102102
35. D. Maturana, S. Scherer, Voxnet: A 3d convolutional neural network for real-time object recognition, in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, (2015), 922–928. https://doi.org/10.1109/IROS.2015.7353481
36. R. Du, Z. Ma, P. Xie, Y. He, H. Cen, PST: Plant segmentation transformer for 3D point clouds of rapeseed plants at the podding stage, *ISPRS J. Photogramm. Remote Sens.*, **195** (2023), 380–392. https://doi.org/10.1016/j.isprsjprs.2022.11.022
37. C. R. Qi, H. Su, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE computer society, Honolulu, HI, USA, (2017), 652–660. https://doi.org/10.48550/arXiv.1612.00593

38. C. R. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, *Adv. Neural Inf. Process. Syst.*, **30** (2017). https://doi.org/10.48550/arXiv.1706.02413
39. H. Kang, H. Zhou, X. Wang, C. Chen, Real-time fruit recognition and grasping estimation for robotic apple harvesting, *Sensors*, **20** (2020), 5670. https://doi.org/10.3390/s20195670
40. T. Masuda, Leaf area estimation by semantic segmentation of point cloud of tomato plants, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, IEEE computer society, Montreal, QC, Canada, (2021), 1381–1389. https://doi.org/10.1109/ICCVW54120.2021.00159
41. D. Li, G. Shi, J. Li, Y. Chen, S. Zhang, S. Xiang, et al., PlantNet: A dual-function point cloud segmentation network for multiple plant species, *ISPRS J. Photogramm. Remote Sens.*, **184** (2022), 243–263. https://doi.org/10.1016/j.isprsjprs.2022.01.007
42. M. Ghahremani, B. Tiddeman, Y. Liu, A. Behera, Orderly disorder in point cloud domain, in *Computer Vision–ECCV 2020: 16th European Conference*, Glasgow, UK, (2020), 494–509. https://doi.org/10.1007/978-3-030-58604-1_30
43. M. Ghahremani, K. Williams, F. M. K. Corke, B. Tiddeman, Y. Liu, J. H. Doonan, Deep segmentation of point clouds of wheat, *Front. Plant Sci.*, **12** (2021), 608732. https://doi.org/10.3389/fpls.2021.608732
44. M. H. Guo, J. X. Cai, Z. N. Liu, T. J. Mu, R. R. Martin, S. M. Hu, Pct: Point cloud transformer, *Comput. Visual Media*, **7** (2021), 187–199. https://doi.org/10.1007/s41095-021-0229-5
45. H. Zhao, L. Jiang, J. Jia, P. H. Torr, V. Koltun, Point transformer, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, IEEE computer society, Montreal, QC, Canada, (2021), 16259–16268. https://doi.org/10.1109/ICCV48922.2021.01595
46. N. Engel, V. Belagiannis, K. Dietmayer, Point transformer, *IEEE Access*, **9** (2021), 134826–134840. https://doi.org/10.1109/ACCESS.2021.3116304
47. J. Lin, M. Rickert, A. Perzylo, A. Knoll, Pctma-net: Point cloud transformer with morphing atlas-based point generation network for dense point cloud completion, in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Prague, Czech Republic, (2021), 5657–5663. https://doi.org/10.1109/IROS51168.2021.9636483
48. L. Hui, H. Yang, M. Cheng, J. Xie, J. Yang, Pyramid point cloud transformer for large-scale place recognition, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, IEEE computer society, Montreal, QC, Canada, (2021), 6098–6107. https://doi.org/10.1109/ICCV48922.2021.00604
49. X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, J. Lu, Point-bert: Pre-training 3d point cloud transformers with masked point modeling, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE computer society, New Orleans, LA, USA, (2022), 19313–19322. https://doi.org/10.48550/arXiv.2111.14819
50. D. Li, J. Li, S. Xiang, A. Pan, PSegNet: Simultaneous semantic and instance segmentation for point clouds of plants, *Plant Phenomics*, **2022** (2022). https://doi.org/10.34133/2022/9787643
51. E. Nezhadarya, E. Taghavi, R. Razani, B. Liu, J. Luo, Adaptive hierarchical down-sampling for point cloud classification, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE computer society, Seattle, WA, USA, (2020), 12956–12964. https://doi.org/10.1109/CVPR42600.2020.01297

52. X. Lai, J. Liu, L. Jiang, L. Wang, H. Zhao, S. Liu, et al., Stratified transformer for 3d point cloud segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE computer society, New Orleans, LA, USA, (2022), 8500–8509. https://doi.org/10.1109/CVPR52688.2022.00831
53. M. Tomkins, Towards modelling emergence in plant systems, *Quant. Plant Biol.*, **4** (2023), e6. https://doi.org/10.1017/qpb.2023.6
54. A. Chaudhury, F. Boudon, C. Godin, 3D plant phenotyping: All you need is labelled point cloud data, in *Computer Vision–ECCV 2020 Workshops*, Glasgow, UK, **16** (2020), 244–260. https://doi.org/10.1007/978-3-030-65414-6_18
55. U. Krämer, Planting molecular functions in an ecological context with Arabidopsis thaliana, *Elife*, **4** (2015), e06100. https://doi.org/10.7554/eLife.06100
56. C. Wyman, The Alias Method for Sampling Discrete Distributions, *Ray Tracing Gems II: Next Generation Real-Time Rendering with DXR, Vulkan, and OptiX*, (2021), 339–343. https://doi.org/10.1007/978-1-4842-7185-8_21
57. S. Laine, T. Karras, Efficient sparse voxel octrees, in *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, Association for Computing Machinery, New York, NY, USA, (2010), 55–63. https://doi.org/10.1145/1730804.1730814
58. Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, et al., Learning semantic segmentation of large-scale point clouds with random sampling, *IEEE Trans. Pattern Anal. Mach. Intell.*, **44** (2021), 8338–8354. https://doi.org/10.1109/TPAMI.2021.3083288
59. L. Li, L. He, J. Gao, X. Han, Psnet: Fast data structuring for hierarchical deep learning on point cloud, *IEEE Trans. Circuits Syst. Video Technol.*, **32** (2022), 6835–6849. https://doi.org/10.1109/TCSVT.2022.3171968
60. R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, et al., On layer normalization in the transformer architecture, in *International Conference on Machine Learning*, Association for Computing Machinery, New York, NY, USA, (2020), 10524–10533. https://doi.org/10.48550/arXiv.2002.04745
61. C. Moenning, N. A. Dodgson, A new point cloud simplification algorithm, in *Proc. Int. Conf. Visualization Imaging Image Proc.*, (2003), 1027–1033.
62. M. Connor, P. Kumar, Fast construction of k-nearest neighbor graphs for point clouds, *IEEE Trans. Visual Comput. Graphics*, **16** (2010), 599–608. https://doi.org/10.1109/TVCG.2010.9
63. J. L. Ba, J. R. Kiros, G. E. Hinton, Layer normalization, Preprint. ArXiv:160706450. https://doi.org/10.48550/arXiv.1607.06450
64. O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference*, Munich, Germany, **18** (2015), 234–241. https://doi.org/10.1007/978-3-319-24574-4_28
65. I. Ziamtsov, K. Faizi, S. Navlakha, Branch-Pipe: Improving graph skeletonization around branch points in 3D point clouds, *Remote Sens.*, **13** (2021), 3802. https://doi.org/10.3390/rs13193802
66. M. Xu, R. Ding, H. Zhao, X. Qi, PAConv: Position adaptive convolution with dynamic kernel assembling on point clouds, in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE computer society, Nashville, TN, USA, (2021), 3172–3181. https://doi.org/10.1109/CVPR46437.2021.00319

67. J. Morel, A. Bac, T. Kanai, Segmentation of unbalanced and in-homogeneous point clouds and its application to 3D scanned trees, *Visual Comput.*, **36** (2020), 2419–2431. https://doi.org/10.1007/s00371-020-01966-7
68. J. Le Louëdec, G. Cielniak, 3D Shape sensing and deep learning-based segmentation of strawberries, *Comput. Electron. Agric.*, **190** (2021), 106374. https://doi.org/10.1016/j.compag.2021.106374
69. H. Weiser, L. Winiwarter, J. Schäfer, F. E. Fassnacht, K. Anders, A. M. E. Pena, et al., Virtual laser scanning (VLS) in forestry-Investigating appropriate 3D forest representations for LiDAR simulations with HELIOS++, in *EGU General Assembly Conference Abstracts*, Vienna, Austria, (2021), EGU21-9178. https://doi.org/10.5194/egusphere-egu21-9178

AIMS Press