Part Segmentation of Synthetic Tree Pointclouds with Deep Learning

Gunner Stone, Sushmita Sarker, Laura Wade, Jonathan Greenberg, and Alireza Tavakkoli

***Abstract*—Terrestrial and Airborne Laser Scanning are effective tools that allow forestry professionals to digitally catalog entire landscapes into large datasets of pointclouds. Extensive work has been performed on tree modeling from isolated tree-pointclouds to estimate topological, geometrical, and volumetric details of the woody structure of a tree. Some of these works perform modeling on rule-based algorithms which are highly dependant on user input to have high quality results. This paper proposes a novel method to accurately and automatically segment tree sections from isolated-tree pointclouds using a synthetic tree pointcloud dataset along with several deep learning pointcloud segmentation models. Firstly, synthetic trees are stochastically generated using SpeedTree. These 3D synthetic tree objects then have their mesh topology extracted and labeled for each point in the pointcloud according to its corresponding L-System Generator. From these semantically labeled pointclouds, several models were trained and evaluated for part segmentation performance: Pointnet, Pointnet++, ShellNet, DGCNN, and PVT. Each model was evaluated on their performance in segmenting each point as belonging to one of three specific tree sections: Trunk, Branch, and Leaf. Although these results are based on synthetic tree pointcloud data, they help to validate a novel method that is an important step forward towards using synthetically created trees using L-Systems as the basis for fine tuning a transfer learning approach to synthetic models with real trees.**

***Index Terms*—L-Systems, TLS, ALS, Pointcloud, Remote Sensing, Tree Part Segmentation, Point Voxel Transformer, PVT.**

I. Introduction

OBSERVING forests and how their ecosystems and biodiversity change over time is an integral component of climate change monitoring and mitigation. Trees have a natural ability to absorb carbon into their biomass and soil. Additionally, uncontrolled and non-prescribed wildfires seek to threaten property, lives, and ecosystem integrity. Forest inventory and analysis are important measures that aid in both quantifying the amount of carbon taken out of the atmosphere and determining fire prevention strategies. Modeling trees for ecological simulation or forestry management can help augment forest inventories by providing better guides for terrestrial studies surrounding fire prevention. Tree models can be used to conduct simulations on how trees combust under certain conditions[1]. Additionally, topographically accurate tree models are able to provide important details for use in allometric equations to estimate above ground biomass (AGB), which is an important determinant of wildfire hazards[2]. High quality tree models are therefore able to help contribute to ecosystem protection and resource preservation.

Traditionally, forest inventorying is conducted manually by professionals, which requires technicians to have extensive training and perform manual data entry. Not only is this process time consuming and error prone, but also costly. Therefore, automating the forest inventoring process would have significant implications for saving expenses in both training time and errors from manual data entry. Currently, a technological solution that aims to take the labor cost away from forest inventorying utilizes terrestrial and airborne laser scanning (TLS/ALS) to perform pointcloud registration and create a small plot of combined LiDAR points. LiDAR has already been utilized in helping determine individual tree assessments, forest structure analysis, and forest typing [3].

Segmentation tasks have been used for a number of robotics applications, including robot manipulation, autonomous navigation, and camera localization, but their application to the segmentation of natural objects, such as trees, into their constituent parts is still an unexplored field of study. This is due to the fact that, in contrast to structures created by humans, natural structures like trees exhibit complex geometry, non-rigidity, recurring parts, non-parametric surfaces, and self-occlusion. Traditional image analysis techniques can be used to perform tree-part segmentation, but this method calls for hand-crafted feature engineering, which is typically restricted to particular settings and fruit trees. Fully convolutional networks (FCN) now dominate modern tree-part segmentation. Common techniques for measuring morphological plant features include two-dimensional (2D) image-based phenotyping methodologies. The drawbacks of these 2D methods are that they cannot account for overlap and occlusion, and because there is no third dimension, it is impossible to accurately estimate size and area. As a result, the learned phenotypic features are incomplete. In order to overcome the limitations of 2D, three-dimensional (3D) imaging is encouraged for phenotyping.

A few related studies have concentrated on tasks like segmenting individual trees from forest point clouds [4], [5], calculating the leaf area density (LAD) of the plant [6], and separating canopy volume from foliage point clouds scanned by a Lidar [7]. The first segmentation of single-lobed leaves from point clouds was carried out by Chaurasia and Beardsley [8] using a superpixel graph clustering technique, and the segmentation outcomes were later improved using

Manuscript created December, 2022; This work was supported in part by the National Science Foundation under Grant No. OIA- 21487. (*Corresponding author: Gunner Stone.*)

Gunner Stone (email: gunnerinlv@gmail.com), Sushmita Sarker, and Alireza Tavakkoli are with the Department of Computer Science and Engineering, University of Nevada, Reno, NV 89557 USA.

Laura Wade and Jonathan Greenberg are with the Department of Natural Resources Environmental Science, University of Nevada, Reno, NV 89557 USA.

Iterative Closest Point (ICP) matching. In a pipeline of four phases, Paproki et al. [9] segmented leaves, petioles, and internodes from a model of Gossypium Hirsutum created from 64 images using multi-view 3D reconstruction and 3D meshes. In their work, Li et al. [10] included a facet over-segmentation and facet region growing-based individual leaf segmentation technique for dense plant point clouds. The spatial features of each point in the point cloud were determined using iterative principal component analysis (IPCA), and facet adjacency was employed to carry out facet region growth using a breadth-first search approach. Several aspects are merged after region expansion to create a larger spatial structure known as a segmented leaf. The results of the experiments demonstrate that the suggested technique is successful and efficient in separating individual leaves from 3D point clouds of greenhouse ornamentals such as Epipremnum Aureum, Monstera deliciosa, and Calathea makoyana. In the work of Shi et al. [11], a method for segmenting 2D plant images using a multi-view approach is provided. The segmented images are then projected into 3D space to create a segmented 3D plant model. To be more precise, semantic and instance segmentation on the 2D images was performed using a fully convolutional network (FCN) and a masked R-CNN. The 3D point cloud was then segmented using various viewpoints. They used PointNet[12], as their baseline network architecture.

As a pioneer, PointNet[12], analyzes point clouds with shared MLPs and further aggregates the data via pooling layers but ignores local relationships between geometric shapes. The effectiveness of the 2D and multi-view approaches was evaluated on tomato seedling plants. Due to PointNet's constraints, the method was only tried on small plants, but it works effectively. Boogaard et al. [13] collected phenotypic data about the cucumber plant's architecture and segmented it into eight parts using a point cloud segmentation model, PointNet++ [14]. Point clouds of plants frequently exhibit a significant level of class imbalance. As a result, the segmentation quality for the underrepresented classes lags behind that of the overrepresented classes in semantic-segmentation approaches. In their other paper, Boogard et al. [15] suggested a method to break up large point clouds into chunks that were independently segmented and then combined later in order to improve the segmentation of underrepresented classes based on the degree of class balance in the training data for the neural network. For the segmentation model, they again used PointNet++ [14], one of the top-performing neural network architectures for point cloud segmentation. PointNet++[14] overcomes the shortcomings of PointNet by leveraging hierarchical spatial structure to boost sensitivity to local geometric arrangement.

In Amatya et al.[16], tree-part segmentation was performed for detection of cherry tree branches to be used in an automated sweet-cherry harvesting vision system. Their implementation utilized both an RGB image-based segmentation method to identify visible segments of branches and then a bayesian classifier to classify images into four classes - branch, cherry, leaf and background. Not utilizing a synthetic dataset, they were forced to constrain their data collection method to have controlled lighting and also had to manually segment pixel-wise class labels. Not accounting for class-distribution, they achieved 89.6, 73.3, 86.9, 91.2% accuracies for branch, cherry, leaf, background classes respectively. Similarly, Lin et al.[17] utilized channel and spatial attention maps with CNN's to perform real-time tree-part segmentation on RGB images of guava trees to segment out fruits from branches, achieving an IoU value of 63.33 and 66.25% for the branches and the fruits. Lin's 891 RGB dataset took them 2 months alone to manually create segmented labels.

More traditional methods, like using pointcloud data to reconstruct quantitative structure models (QSM) for trees, show promise in AGB estimation and part-segmentation[18]. However, these methods require someone proficient in manually fine-tuning the algorithm's input parameters to produce quality cylindrical estimates of the tree's topology. Depending on the size and complexity of the input tree, QSM may also take a large inference time to calculate and produce results[19], which may not be feasible in an application at-scale.

Despite the fact that PointNet and PointNet++ pioneered point cloud learning, there are some other benchmark models that showed better performance. To conduct a comparative analysis, we have assessed the results of various cutting-edge models in addition to PointNet and PointNet++.

A number of approaches connect the point set into a graph and conduct message passing on this graph. In DGCNN[20], an EdgeConv is proposed to produce edge features that explain the connections between a point and its neighbors. This allows for the construction of a local graph while maintaining point associations. EdgeConv can group points in both Euclidean and semantic space since it explicitly builds a local graph and learns the embeddings for the edges. In order to efficiently execute point neighbors and convolution with points, the authors of ShellNet[21] proposed shellconv as the primary convolution operator to divide a local point neighborhood. In order to do this, they divide each point's neighborhood into concentric spherical shells by querying the neighborhood at each point and applying a set of concentric spheres to each partition. Based on the statistics of the points inside, it is possible to extract representative features from each shell. Voxel-based models are capable of efficiently encoding coarse-grained characteristics and have regular data locality. However, due to their flexible fields, point-based networks maintain the accuracy of position information. Inspired by this, Zhang et al.[22] proposed a hybrid point cloud learning architecture called PointVoxel Transformer, which combines the concepts of voxel-based and point-based models. The authors collected coarse-grained local features from nonempty voxels using the Sparse Window Attention (SWA) module. The module obtains linear computational complexity with respect to voxel resolution without the need of doing costly irregular data organization and erroneous empty voxel calculation. The authors also employed two distinct self-attention versions in order to collect fine-grained details about the global shape.

Many of these works highlight that one of limitation they encounter is that they would achieve higher quality results if they had larger quantities of pointcloud training data for their models to use. As already mentioned, gathering real-world labeled pointcloud data to be used in these models is

very time consuming, costly, and requires expert-level domain knowledge. Because of these factors, a large publicly available dataset to help alleviate these issues is not available for researchers to collaborate with. This paper turns to synthetically created pointcloud data to see if it can help bridge this data-shortage gap.

Realistic looking tree models that aren't derived from TLS/ALS measurements are commonly used in urban landscape design and virtual entertainment industries. Popular middleware such as SpeedTree and Houdini are used in these applications to allow designers to produce realistic tree models with relative ease. Under the hood, these middlewares use stochastic parameterized context-sensitive Lindenmayer Systems (L-Systems) to drive the rules that procedurally generate their synthetic trees.

L-Systems were originally developed in 1968 by Aristid Lindenmayer as a way to mathematically model the growth processes of plant development [23]. They have since been heavily extended, and are currently used in the film and video game industry to procedurally generate realistic looking synthetic trees [24], [25]. Each synthetic tree is built using a parametric set of recursive production rules that describe how to build a tree for a given number of steps. By changing the parameter values of a given L-System, trees that have significantly different visual characteristics and structural features can be produced.

Digumarti et al.[26] utilized SpeedTree in a part-segmentation task to generate 6 synthetic broadleaf tree species with 5 instances each. They used Unreal Engine to help simulate a real world robot image-capturing scenario. Their robot captured RGB-D images from different angles as it traveled around the synthetic tree in a spiral motion. Not accounting for wind, and isolating their target tree by itself in a scene, they concurrently trained several models. Some models utilized both the RGB and HHA channels that their synthetic robot captured. Their SegNet[27]-based autoencoder, named RGB-net, embedded an RGB image to a set of feature-vectors and an auxiliary model called HHA(HA)-net generated a feature embedding from the HHA-encoded depth channels. Both these modality embeddings were then concatenated and fed into a simple CNN to produce a segmented 2D image containing 6 classes (trunk, branch, twig, leaf, ground, sky). Their model attained a class weighted total accuracy of 0.867% and an unweighted accuracy of 0.925%. This paper then performed a qualitative evaluation on real-world data to see if their model, which trained on synthetic trees, was transferable to the real-world domain. While no quantitative results were given because of the absence of labelled datasets of vegetation, they claim their results were meaningful upon manual examination and showed that their models were able to generalize to a real-world setting.

Extending upon the work done by previous literature, this proposed work provides the following major contributions:

- A novel pipeline for generating a sufficiently large pointcloud dataset of synthetic trees for use in a deep learning task.
- A comparative analysis between 5 pointcloud part-segmentation deep learning models, trained using the synthetic dataset to perform tree part-segmentation.

Harnessing the procedural modeling of L-Systems using SpeedTree, coupled with a novel reoptimization pipeline, realistic pointclouds of tree models can be stochastically generated to create large synthetic pointcloud datasets. Utilizing a synthetic dataset, a model can then be trained to solve a 3 class part-segmentation task on a tree's trunk, branches, and leaves only using a pointcloud containing the Cartesian coordinates (*x, y, z*) of each point. The model, having only been originally trained on synthetic trees, can later be used to validate against real data, collected from TLS/ALS pointclouds of isolated trees.

II. Methodology

A. Selecting L-System Seeds

Data was obtained using the SpeedTree Model Library Store[28]. From a list of seven target species in the Pacific Northwest (quaking aspen, Douglas fir, western white pine, black oak, western juniper, ponderosa pine, and Jeffrey pine), high quality models existed for 5 of the 7. At present, there were not any sufficiently high quality realistic looking models for ponderosa pine and Jeffrey pine, which is why they were omitted from this dataset. These five dominant species were selected by an expert ecologist based on how well each resemble their real-life counterparts. This design choice was made in order for model generalizability and transferability to datasets containing LIDaR scans of real trees. Each species model contained the L-System parameterization values needed to stochastically generate other, synthetic, look-a-like copies of the original 'seed' tree. Samples of stochastic generations can be visualized in Fig 1. This study's results and discussion utilize models only trained using synthetic trees produced from the quaking aspen L-System 'seed'.

B. Reoptimizing Tree Models

To prepare these models, whose polygon counts are optimized for use in movies and videogames, they first have to be reoptimized to be useful in the context of a pointcloud dataset. Each vertex in an exported SpeedTree object becomes a point in the pointcloud sample, so it is important to have a realistic distribution of vertex/points similar to a LIDaR scan. SpeedTree Generators can be thought of as the parametric rules of an L-System[23] that determine how Nodes are placed and what each Node looks like. Generators contain hundreds of unique parameters that each contribute to the overall structure of the tree. Editing a combination of these parameter values can significantly change a tree's topology; however, certain parameters only affect the computational fidelity of the underlying polygon mesh. The reoptimization process aims at increasing both the raw number and density of vertices along Nodes in the tree's polygon mesh. Increasing the length and radial absolute values of a Generator's segment will increase vertex count and density along Nodes produced by the Generator. Every Generator also has an Optimization parameter, who's value should also be set to the minimum. Meshes in the tree model, which are often used for leaves, fronds, and dead branches, should also have reoptimization

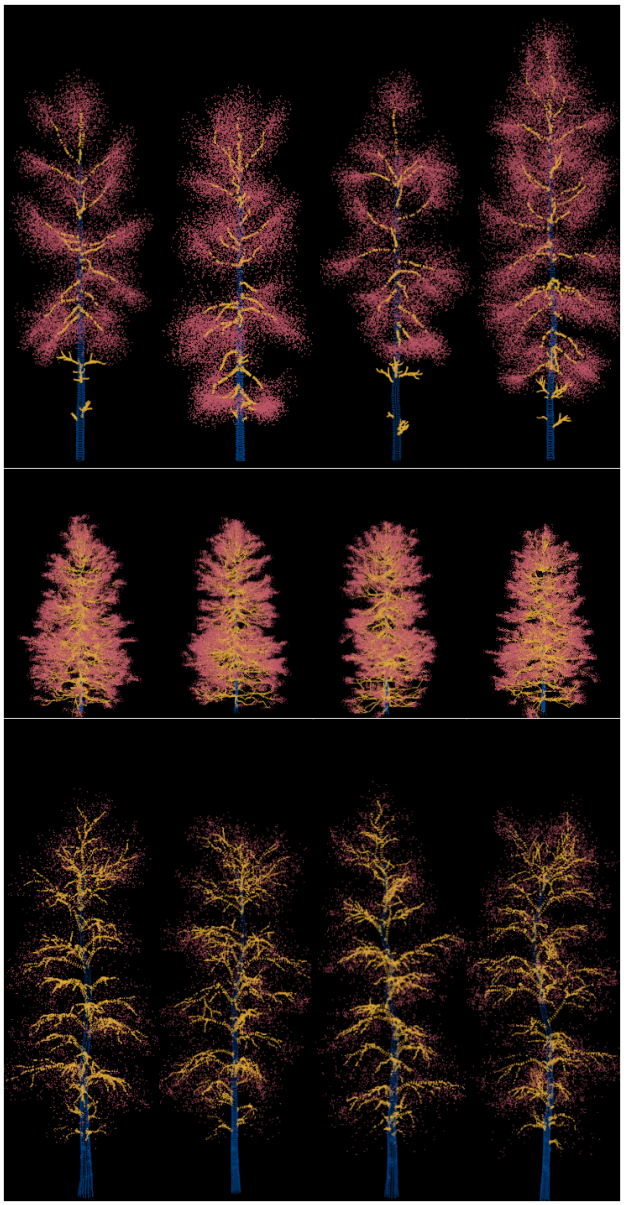

Fig. 1. Visualization of pointclouds generated for quaking aspen (top), Douglas fir (middle), and white pine (bottom). Each model is generated using the same species' parametric L-System seed. Adding stochastic noise to these L-System seeds helps create structurally different trees. The color of points in the pointcloud

performed by adding additional anchor points to the mesh, which function similarly to vertices in the exported model. A tree model having undergone this process is illustrated in Fig.2, where its raw point count has been increased by around 25x. Each species of tree will manually undergo this reoptimization process, as each separate SpeedTree model may have more or less Generators that need to be individually repotimized.

C. Exporting Ground Truth Point Labels

SpeedTree's custom exporter exports the complete mesh-topology of a given tree as a hierarchical XML document. The schema of the document similarly represents the hierarchical structure of the Generator topology of the original tree. This

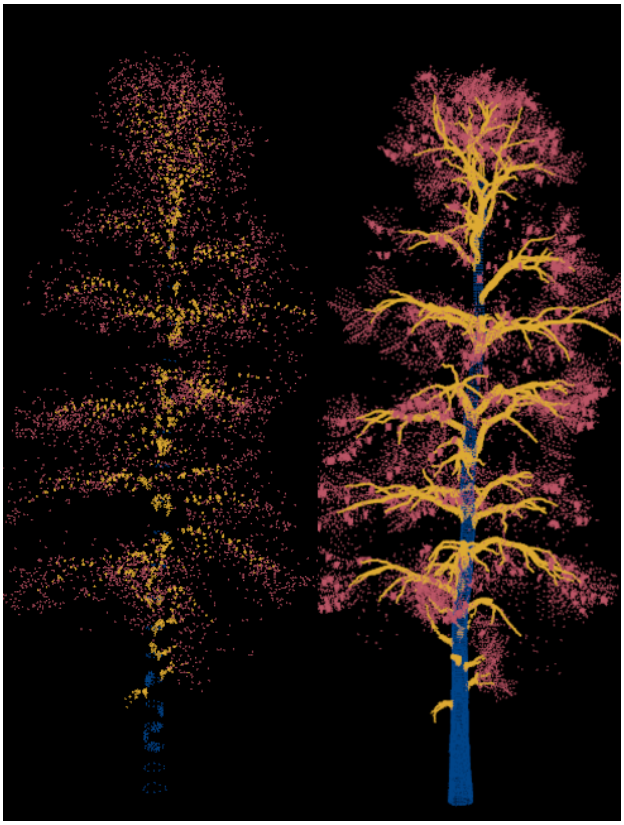

Fig. 2. A synthetic white pine model without pointcloud reoptimization performed (left) and with reoptimization performed (right). Without reoptimization the pointcloud appears visually sparse and contains only 11,483 points. By applying reoptimization it contains 283,649 points.

means that vertices from the same Node are stored together, allowing for groups of points to maintain their original semantic label throughout the exportation process. By strategically self-labeling Generators in the original SpeedTree Model, it is possible to export a pointcloud of ground-truth semantic labels with Node-wise granularity.

D. Dataset Preparation

After the unoptimization process, 1000 stochastic generations were exported for each species composing a total of 5000 unique tree pointclouds. Each tree pointcloud also contains a class label for every point as belonging to a trunk, branch, or leaf Node. This dataset then underwent an 80/10/10 split on a per-species basis for training, validation, and testing. Following other deep learning standards for point clouds, each training sample underwent several preprocessing steps to achieve data normalization and data augmentation. The general model training pipeline demonstrating this process is shown at a high-level in Fig. 3.

Class-weighted random subsamplings were first applied to downscale the pointclouds to 2048 points. Downsampling pointclouds achieves a two-fold goal. Firstly, downsampling performs data augmentation in a way that helps to enrich the data diversity that the model receives as input, allowing the same original pointcloud to be reused multiple times during the training process without the model memorizing specific point

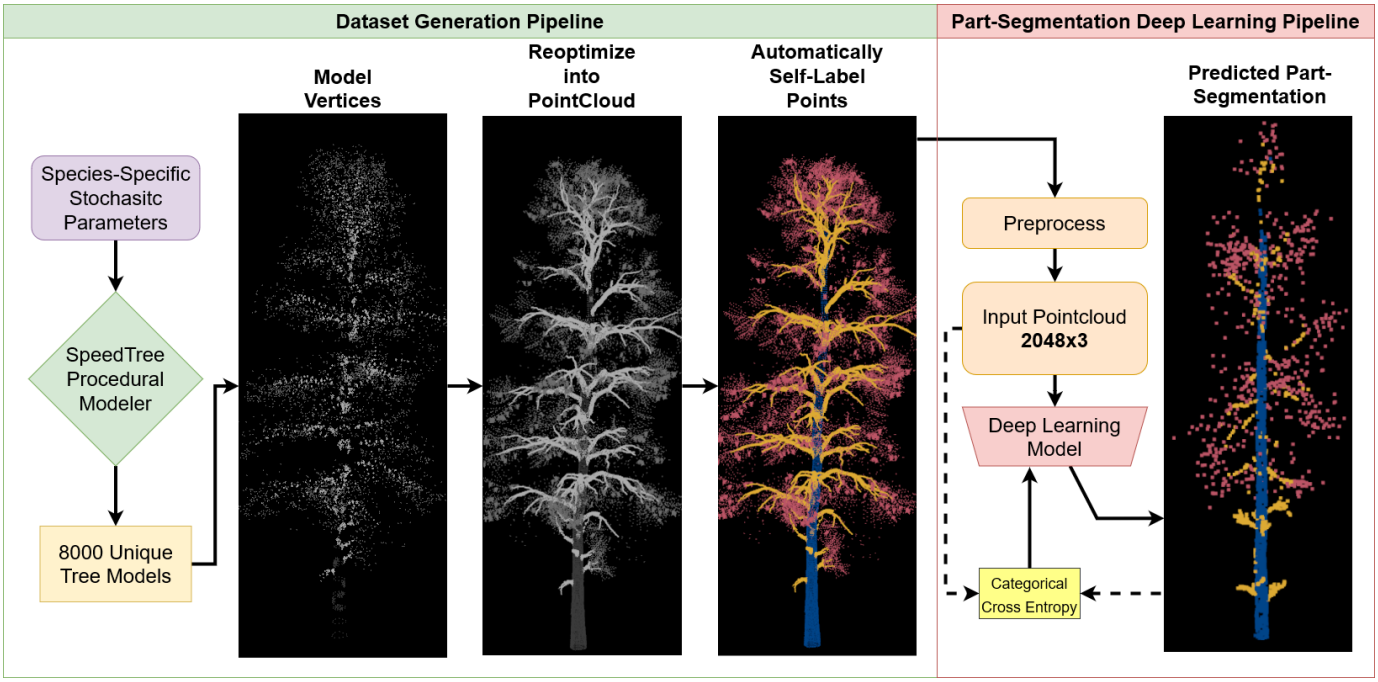

Fig. 3. Pipeline showing the steps to transform the original SpeedTree object into a preprocessed subsampled input ready for model inference. The Deep Learning Model block separately represents each of the five investigated models for this part-segmentation task. Each model uses categorical cross-entropy as the loss function responsible for driving the models to achieve optimal performance.

layouts. Secondly, while input size-invariant deep learning models for pointclouds exist, it has been shown that performing a donwsampling to 2048 points increases model performance on semantic and classification-based tasks[29][22]. Random downsampling was chosen because it makes no assumptions about the input pointcloud space and is relatively fast to perform.

After downsampling, the pointclouds then have a random rotation applied as a data augmentation step in order to help the model further generalize to real-life trees that are not perfectly level with the ground. Lastly, each pointcloud was normalized to fit within a Unit-Sphere to help constrain the feature-space to lie within a common boundary.

E. General Model Frameworks

A total of 5 deep learning models were investigated and evaluated for their performance on a 3-class (trunk, branch, leaf) part-segmentation task for an isolated tree pointcloud. These pointcloud part-segmentation model architectures included: PointNet[12], PointNet++[14], ShellNet[21], Dynamic Graph CNN (DGCNN)[21], and Point-Voxel Transformer (PVT)[22]. Each model's architecture is illustrated at a high-level in Fig. 4. These particular models were chosen based on select criteria: their performance on part-segmentation tasks using popular pointcloud datasets such as ModelNet40[30] and ShapeNet[31], their popularity and acceptability in the pointcloud deep learning community as both baselines and cutting edge architectures, and to compare how different methods, such as hierarchical, voxel, graph, and hybrid, might be better suited for the domain of tree topology.

Each model used the same data-splits for training, validation, and testing for a reproducible comparative analysis of model performance. The loss of each model was calculated using class-weighted categorical cross-entropy. Model performance is quantified using the Dice-coefficient between the predictions and ground truth to account for the imbalanced class distribution of the labeled points. Additionally, part-average Intersection-over-Union (pIoU) is included to help compare model performance on other datasets. After preprocessing, each model takes only the geometric coordinates (x, y, z) of subsampled points as inputs. Each model's hyperparameters were optimized using a Bayesian search based on model performance. All deep learning models were programmed using the GPU version of PyTorch using a hardware configuration with Intel® Xeon® W-2295 18 x 3.0GHz, 512GB RAM, and NVIDIA Quadro P2200 (5GB).

III. Results and Discussion

A. Quantitative and Qualitative Results

Quantitative assessment of part-segmentation is summarized in Table I and visualized in Fig 5. Each model's parameters were saved upon validation convergence, which was gauged when the loss on the validation set stopped increasing and became stable after 3 epochs. In this study, a high testing pIoU (average unweighted class accuracy), high Dice coefficient (weighted class accuracy), and low inference time (model scalability) were all considered as the criteria of a good classifier.

Impressively, PVT was able to significantly outperform every other model in all 3 class domains. This is because of its hybrid-based PVT blocks that simultaneously learn from varying scales of the raw pointcloud and voxelized representations. It aims to learn both local features in the voxel-

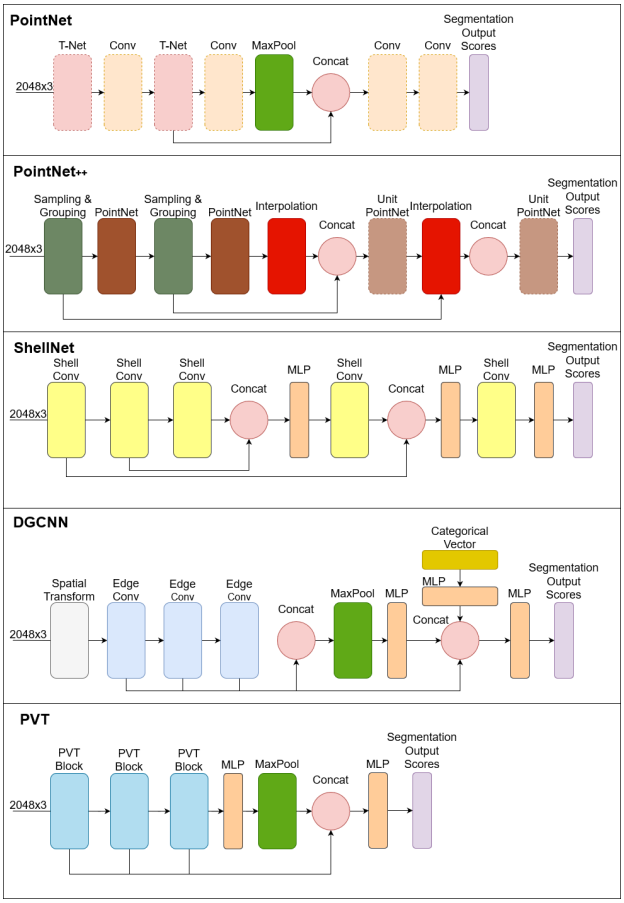

Fig. 4. High-level visualization of the network architecture for each of the deep learning models utilized for tree part-segmentation. Each model in their original paper typically has a dual branching structure to perform both classification and segmentation. Only the segmentation branch is displayed here for both relevance and simplicity.

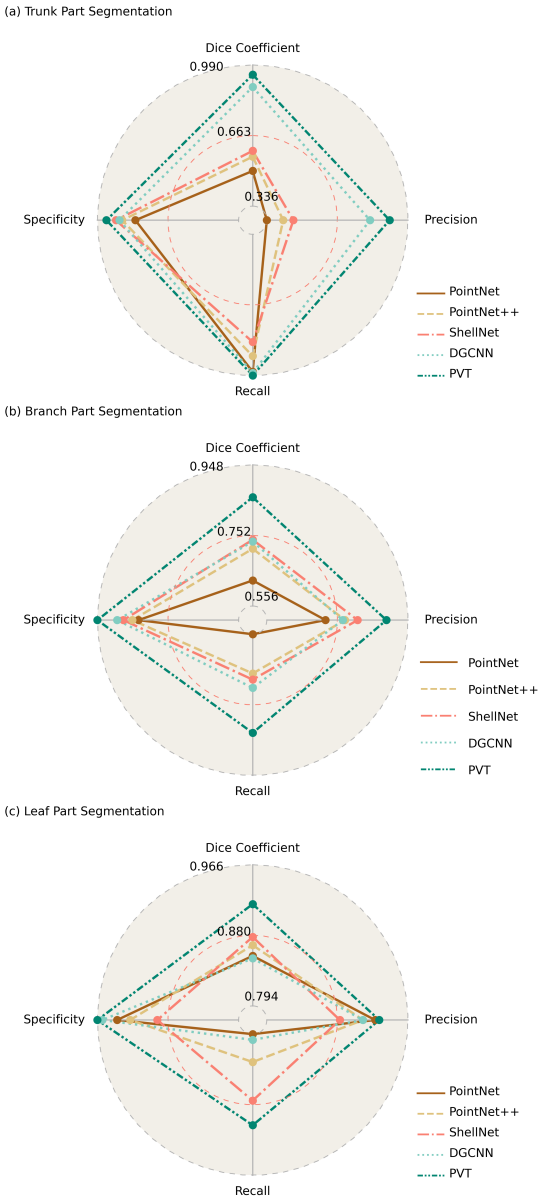

Fig. 5. Visualization of performance using radar charts for each part segmentation task: (a) Trunk, (b) Branch, and (c) Leaf. Each chart compares the Dice Coefficient, Precision, Specificity, and Recall for each deep learning model. Values are normalized such that the minimum value lies at the center ring of the chart and the max value lies on the outermost ring. Each ring is labeled with its corresponding value.

domain and global features in the point-domain. Additionally, it employs a UNET-like architecture[32], which is notable for their impressive performance in 2D segmentation tasks.

In addition, comparing model pIoU performance between this task and ShapeNet part-segmentation, it is interesting to see that all classifiers except for PVT had worse pIoU values. It stands to reason that objects from the ShapeNet dataset are easier for deep learning models to learn about than the complex branching structure of a tree pointcloud.

DGCNN was the most rapid on model inference for a single tree (59 ms), nearly halfing the inference time of PVT (93 ms). DGCNN does not have to voxelize a pointcloud several times during model inference, which could help explain why it sees such performance boost in inference time. Additionally, its Edge Conv blocks seem to perform well on the cylindrical structure of the tree pointcloud. However, this model notably has a limiting hyperparameter K, which determines the number of edge connections a singular point can have. While one value of K might work well for classifying one class, it may limit performance on another class. This would help explain the large value gap in Dice coefficient between the trunk (0.888), branch (0.735), and leaf (0.852).

Comparing the precision and recall values among classes, all models had higher recalls than precisions for trunk, and the inverse for both branch and leaf. This indicates that each model was often over-estimating the number of trunk points. While viewing rendered model predictions, points near the intersection of branches and the trunk were often observed to be the ones misclassified as a trunk. This also might be a difficult problem for humans, so it makes sense why these models had a hard time classifying these ambigous intersection regions.

PointNet++ had the worst model inference time (1113 ms). This large inference time is by design, as the PointNet++ architecture applies PointNet recursively on a nested partitioning of the input point set. By designing the architecture that

TABLE I
STATISTICAL SUMMARY OF THE FIVE TREE PART-SEGMENTATION MODELS SHOWING AVERAGED PERFORMANCE METRICS WHEN EVALUATED ON THE TEST-SET. PIOU MEANS PART-AVERAGE INTERSECTION-OVER-UNION.

		Trunk	**Branch**	**Leaf**
PointNet	Dice Coefficient	0.499	0.627	0.855
	Precision	0.336	0.719	0.926
	Recall	0.973	0.556	0.794
	Specificity	0.814	0.834	0.942
	pIoU	0.773		
	Inference (ms)	206		
	Parameters (m)	3.536		
PointNet++	Dice Coefficient	0.565	0.715	0.868
	Precision	0.412	0.773	0.912
	Recall	0.900	0.665	0.828
	Specificity	0.875	0.851	0.926
	pIoU	0.800		
	Inference (ms)	1113		
	Parameters (m)	0.966		
ShellNet	Dice Coefficient	0.592	0.740	0.878
	Precision	0.459	0.808	0.883
	Recall	0.835	0.682	0.875
	Specificity	0.904	0.876	0.893
	pIoU	0.796		
	Inference (ms)	385		
	Parameters (m)	0.775		
DGCNN	Dice Coefficient	0.888	0.735	0.852
	Precision	0.814	0.767	0.911
	Recall	0.978	0.705	0.801
	Specificity	0.888	0.893	0.960
	pIoU	0.830		
	Inference (ms)	**59**		
	Parameters (m)	1.454		
PVT	Dice Coefficient	**0.945**	**0.858**	**0.918**
	Precision	0.905	0.888	0.931
	Recall	0.990	0.830	0.905
	Specificity	0.948	0.948	0.966
	pIoU	**0.910**		
	Inference (ms)	93		
	Parameters (m)	6.414		

way, PointNet++ is able to learn local features with increasing contextual scales. One of ShellNet's large contributions to the 3D deep learning space was to achieve better performance than PointNet++ while simultaneously speeding up model inference time and decreasing model size. Using ShellConv blocks instead of PointNet blocks helps to drastically speed up model inference time. Additionally, its concentric spherical shells to define representative features helps it closely match the performance of PointNet++ accross the board.

B. Limitations and Future Work

It should be noted that these models, especially PVT, performed well on part-segmentation for subsets of pointclouds. However, depending on the application, more of the remaining pointcloud may need to be automatically labeled. While these remaining points might be predicted naively using K-clustering using each labeled point as a cluster center, this implementation sees significant performance loss when applied to the test-dataset. A semi-supervised approach may be appropriate for this task. For example, given a 'mostly-correct' labeled subset of a tree using the proposed methods in this paper, train another model to utilize these pre-labels and learn the correct mappings for the rest of the unlabeled pointcloud.

A call for more environmental artists with backgrounds in forest ecology would be helpful in generating more realistic looking synthetic trees. This work's pipeline is limited by what SpeedTree already has in their store library. By bringing talented artists into this research space, more procedural models can be produced and catered towards various domain-specific deep learning tasks.

Another direction for future work may involve the investigation of methods in which these synthetic tree pointclouds are subsampled. Random subsampling is a naive, but fast, approach that may not subsample the pointcloud in an optimal way for the models to learn. Lin et al.[33] have shown that training models to perform a data-driven sampler learning strategy helps to increase base-line model performance for point-wise analysis and segmentation-based tasks.

IV. CONCLUSION

Terrestrial and Airborne Laser Scanning are effective tools that allow forestry professionals to digitally catalog entire landscapes into large datasets of pointclouds. Topographically accurate synthetic tree models, once robustly generated using the proposed method's automatic tree part-segmentation, have the potential to be implemented at-scale. The model-provided information about a given tree's topology contains important details used in allometric equations to estimate above ground biomass (AGB), which is an important determinant of wildfire hazards. This paper's novel contributions include extending the modality of highly accurate tree part-segmentation from 2D (RGB/RGB-D) into the raw, unstructured geometric-coordinate pointcloud domain. Additionally, this paper's novelty serves as a synthetic vegetation pointcloud dataset-generation pipeline for use in training other deep learning remote-sensing tasks like: tree-segmentation, estimating mensuration data, species identification, and wood filtering.

REFERENCES

[1] S. Pirk, M. Jarzabek, T. Hadrich, D. Michels, and W. Palubicki, "Interactive wood combustion for botanical tree models," *ACM Transactions on Graphics*, vol. 36, pp. 1–12, 11 2017.

[2] S. O, X. Hou, and R. Orth, "Observational evidence of wildfire-promoting soil moisture anomalies," *Scientific Reports*, vol. 10, no. 1, Jul. 2020. [Online]. Available: https://doi.org/10.1038/s41598-020-67530-4

[3] D. L. Evans, S. D. Roberts, and R. C. Parker, "Lidar a new tool for forest measurements?" *The Forestry Chronicle*, vol. 82, no. 2, pp. 211–218, 2006. [Online]. Available: https://doi.org/10.5558/tfc82211-2

[4] B. Yang, W. Dai, Z. Dong, and Y. Liu, "Automatic forest mapping at individual tree levels from terrestrial laser scanning point clouds with a hierarchical minimum cut method," *Remote Sensing*, vol. 8, no. 5, p. 372, 2016.

[5] L. Shengyang, L. Zhiwen, L. Kang, and Z. Zifei, "Advances in application of space hyperspectral remote sensing," , vol. 48, no. 3, pp. 303 001–0 303 001, 2019.

[6] S. Li, L. Dai, H. Wang, Y. Wang, Z. He, and S. Lin, "Estimating leaf area density of individual trees using the point cloud segmentation of terrestrial lidar data and a voxel-based model," *Remote sensing*, vol. 9, no. 11, p. 1202, 2017.

[7] R. Ferrara, S. G. Virdis, A. Ventura, T. Ghisu, P. Duce, and G. Pellizzaro, "An automated approach for wood-leaf separation from terrestrial lidar point clouds using the density based clustering algorithm dbscan," *Agricultural and forest meteorology*, vol. 262, pp. 434–444, 2018.

[8] G. Chaurasia and P. Beardsley, "Editable parametric dense foliage from 3d capture," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5305–5314.

[9] A. Paproki, X. Sirault, S. Berry, R. Furbank, and J. Fripp, "A novel mesh processing based technique for 3d plant analysis," *BMC plant biology*, vol. 12, no. 1, pp. 1–13, 2012.

[10] D. Li, Y. Cao, X.-s. Tang, S. Yan, and X. Cai, "Leaf segmentation on dense plant point clouds with facet region growing," *Sensors*, vol. 18, no. 11, p. 3625, 2018.

[11] W. Shi, R. van de Zedde, H. Jiang, and G. Kootstra, "Plant-part segmentation using deep learning and multi-view vision," *Biosystems Engineering*, vol. 187, pp. 81–95, 2019.

[12] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *CoRR*, vol. abs/1612.00593, 2016. [Online]. Available: http://arxiv.org/abs/1612.00593

[13] F. P. Boogaard, E. J. van Henten, and G. Kootstra, "Boosting plant-part segmentation of cucumber plants by enriching incomplete 3d point clouds with spectral data," *biosystems engineering*, vol. 211, pp. 167–182, 2021.

[14] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *CoRR*, vol. abs/1706.02413, 2017. [Online]. Available: http://arxiv.org/abs/1706.02413

[15] F. P. Boogaard, E. J. Van Henten, and G. Kootstra, "Improved point-cloud segmentation for plant phenotyping through class-dependent sampling of training data to battle class imbalance." *Frontiers in plant science*, vol. 13, pp. 838 190–838 190, 2022.

[16] S. Amatya, M. Karkee, A. Gongal, Q. Zhang, and M. D. Whiting, "Detection of cherry tree branches with full foliage in planar architecture for automated sweet-cherry harvesting," *Biosystems Engineering*, vol. 146, pp. 3–15, 2016, special Issue: Advances in Robotic Agriculture for Crops. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1537511015001683

[17] G. Lin, C. Wang, Y. Xu, M. Wang, Z. Zhang, and L. Zhu, "Real-time guava tree-part segmentation using fully convolutional network with channel and spatial attention," *Front. Plant Sci.*, vol. 13, p. 991487, Sep. 2022.

[18] M. I. Disney, M. Boni Vicari, A. Burt, K. Calders, S. L. Lewis, P. Raumonen, and P. Wilkes, "Weighing trees with lasers: advances, challenges and opportunities," *Interface Focus*, vol. 8, no. 2, p. 20170048, 2018. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rsfs.2017.0048

[19] M. Ã…kerblom, P. Raumonen, E. Casella, M. I. Disney, F. M. Danson, R. Gaulton, L. A. Schofield, and M. Kaasalainen, "Non-intersecting leaf insertion algorithm for tree structure models," *Interface Focus*, vol. 8, no. 2, p. 20170045, 2018. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rsfs.2017.0045

[20] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *CoRR*, vol. abs/1801.07829, 2018. [Online]. Available: http://arxiv.org/abs/1801.07829

[21] Z. Zhang, B. Hua, and S. Yeung, "Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics," *CoRR*, vol. abs/1908.06295, 2019. [Online]. Available: http://arxiv.org/abs/1908.06295

[22] C. Zhang, H. Wan, S. Liu, X. Shen, and Z. Wu, "Point-voxel transformer: An efficient approach to 3d deep learning," *CoRR*, vol. abs/2108.06076, 2021. [Online]. Available: https://arxiv.org/abs/2108.06076

[23] A. Lindenmayer, "Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs," *Journal of Theoretical Biology*, vol. 18, no. 3, pp. 300–315, Jan. 1968.

[24] T. Reid, "Hollywood's movie tech wizards honoured by oscars organizers," Feb 2015. [Online]. Available: https://www.livemint.com/Consumer/Jh1N22EQWOzbwbjcU8jORI/Hollywoods-movie-tech-wizards-honoured-by-Oscars-organizers.html

[25] "Speedtree brings photoreal vegetation to the wolf of wall street," January 2014. [Online]. Available: https://www.cgw.com/Press-Center/Web-Exclusives/2014/SpeedTree-Brings-Photoreal-Vegetation-to-The-Wol.aspx

[26] S. Tejaswi Digumarti, L. M. Schmid, G. M. Rizzi, J. Nieto, R. Siegwart, P. Beardsley, and C. Cadena, "An approach for semantic segmentation of tree-like vegetation," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 1801–1807.

[27] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *CoRR*, vol. abs/1511.00561, 2015. [Online]. Available: http://arxiv.org/abs/1511.00561

[28] I. Interactive Data Visualization, *SpeedTree Cinema*, ver. 8, Lexington, SC, USA, 2022 [Online]. [Online]. Available: https://store.speedtree.com/

[29] Z. Xi, C. Hopkinson, S. B. Rood, and D. R. Peddle, "See the forest and the trees: effective machine and deep learning algorithms for wood filtering and tree species classification from terrestrial laser scanning," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 168, pp. 1–16, 2020.

[30] Z. Wu, S. Song, A. Khosla, X. Tang, and J. Xiao, "3d shapenets for 2.5d object recognition and next-best-view prediction," *CoRR*, vol. abs/1406.5670, 2014. [Online]. Available: http://arxiv.org/abs/1406.5670

[31] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "Shapenet: An information-rich 3d model repository," *CoRR*, vol. abs/1512.03012, 2015. [Online]. Available: http://arxiv.org/abs/1512.03012

[32] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[33] Y. Lin, L. Chen, H. Huang, C. Ma, X. Han, and S. Cui, "Beyond farthest point sampling in point-wise analysis," *CoRR*, vol. abs/2107.04291, 2021. [Online]. Available: https://arxiv.org/abs/2107.04291