Tree Instance Segmentation in Urban 3D Point Clouds Using a Coarse-to-Fine Algorithm Based on Semantic Segmentation

Josafat-Mattias Burmeister[1], Rico Richter[1], Stefan Reder[2], Jan-Peter Mund[2], Jürgen Döllner[3]

[1] University of Potsdam, Digital Engineering Faculty, Potsdam, Germany
(burmeister, rico.richter.1)@uni-potsdam.de
[2] Eberswalde University for Sustainable Development, Faculty of Forest and Environment, Eberswalde, Germany
(stefan.reder, jan-peter.mund)@hnee.de
[3] University of Potsdam, Digital Engineering Faculty, Hasso Plattner Institute, Potsdam, Germany
doellner@uni-potsdam.de

Keywords: 3D Point Clouds, Tree Instance Segmentation, Deep Learning, Urban Forestry, Tree Inventory, LiDAR

Abstract

3D point clouds acquired with terrestrial or mobile LiDAR sensors are increasingly used to map urban forests. The segmentation of separate tree instances, i.e., subsets of points representing individual trees, is a relevant step in automatically extracting tree inventory data from 3D point clouds. Various algorithms have been proposed for tree instance segmentation, offering different trade-offs between accuracy, runtime, and robustness against data incompleteness and noise. In this work, we propose a coarse-to-fine algorithm for segmenting tree instances in urban 3D point clouds from terrestrial or mobile LiDAR scanning that combines two existing techniques: (1) the computationally efficient marker-controlled Watershed algorithm and (2) a more accurate 3D region growing algorithm. Initially, the marker-controlled Watershed algorithm generates a coarse segmentation, which is further improved by a Voronoi segmentation-based error removal. Subsequently, the coarse segmentation is refined by the 3D region growing algorithm in areas with overlapping tree crowns and sufficient data quality. In both steps, our algorithm uses the results of a prior semantic segmentation to select suitable markers and seed points. We evaluated our coarse-to-fine algorithm in an ablation study using two mobile LiDAR datasets and one terrestrial LiDAR dataset from three German cities. Our results show that our algorithm outperforms the standard marker-controlled Watershed algorithm in terms of panoptic quality by 3.7, 25.5, and 29.6 percentage points, respectively, while being computationally more efficient than an approach purely based on 3D region growing.

1. Introduction

Urban forests, i.e., the trees growing in a city, provide a wide range of ecosystem services and therefore play a crucial role in sustainable urban development (Säumel et al., 2016; Endreny, 2018). At the same time, they must also meet traffic safety requirements (Bennett, 2010) and compete for space with buildings and infrastructure (Yazdi et al., 2023). Tree cadastres, which record the location and selected attributes of individual trees, serve as the basis for the management of urban trees (Kjeldsen-Kragh Keller and Konijnendijk, 2012). Since manually collecting such tree inventory data is laborious, the use of LiDAR systems to map urban trees has been explored in recent years (Chen et al., 2019). Specifically, terrestrial laser scanning (TLS) with stationary sensors (Yazdi et al., 2024), mobile laser scanning (MLS) with vehicle-mounted sensors (Herrero-Huerta et al., 2018), and personal laser scanning (PLS) with handheld sensors (Comesaña-Cebral et al., 2021) produce high-resolution side-view 3D point clouds that can be used to derive dendrometric attributes, such as trunk diameter at breast height, tree height, or crown volume. 3D point clouds acquired in urban environments typically cover multiple trees and their surroundings, such as the ground, buildings, and city furniture. To obtain measurements of individual trees, it is necessary to isolate subsets of points representing individual trees. This involves separating tree points from non-tree points, which is a semantic segmentation task, and dividing the set of tree points into subsets representing individual trees, which is an instance segmentation task. Although there are methods that accomplish both tasks in a single processing step, known as panoptic segmentation (Xiang et al., 2023), our work specifically focuses on the instance segmentation task of individual tree segmentation (ITS). We assume that semantic segmentation information is already available. For this, we build on the semantic segmentation approach of Burmeister et al. (2023), which classifies points as representing the ground, low vegetation, tree trunks, tree branches, or tree crowns using a deep learning (DL) model. Furthermore, we focus on side-view 3D point clouds of urban vegetation stands acquired by LiDAR scanners from below the canopy. Several ITS approaches have been proposed for this setting, including clustering and region growing approaches (Li et al., 2021), graph-based approaches (Hui et al., 2022), as well as DL approaches (Luo et al., 2021). However, existing approaches have the following limitations:

1. Most previously published algorithmic methods have limited robustness against occlusions and noise, as they often rely on detecting specific parts of a tree, e.g., the trunk or the top of the crown.

2. DL methods for instance segmentation can capture more complex data patterns and therefore are a promising approach for ITS. However, their training is computationally expensive and requires a large amount of annotated training data, whose creation is laborious.

3. Existing methods do not dynamically adapt to varying tree densities and data quality. On the one hand, there are algorithms that assume that the crowns of neighboring trees do not overlap, or that operate at a coarse level of data resolution, leading to inaccuracies in dense vegetation stands.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-4-W5-2024-79-2024 | © Author(s) 2024. CC BY 4.0 License.

On the other hand, there are algorithms that involve computationally expensive steps to separate overlapping tree crowns, resulting in an unnecessarily high runtime when tree crowns do not overlap or when the data quality is too low to benefit from more accurate segmentation techniques.

To address the above issues, we propose a coarse-to-fine ITS algorithm, which incorporates information from semantic segmentation. Using semantic information, our algorithm can localize trees based on both tree trunks and crown tops, increasing its robustness to occlusions. Our approach relies on a DL model only for semantic segmentation, thus it does not require training data with tree instance labels, which are more laborious to create than semantic segmentation labels when tree crowns overlap significantly. Through a coarse-to-fine approach, our algorithm dynamically adapts to varying tree densities and data quality. In the first step, a coarse segmentation is obtained using the computationally efficient marker-controlled Watershed algorithm (Kornilov and Safonov, 2018). In areas with overlapping tree crowns and sufficient data quality, the segmentation is refined by a more computationally expensive but more accurate 3D region growing algorithm. Our algorithm is evaluated on manually annotated MLS and TLS datasets from three cities.

2. Related Work

A wide range of ITS approaches is described in the literature, often specializing in data from specific habitat types (e.g., urban environments, natural forests) or acquisition methods (e.g., TLS, airborne laser scanning (ALS)). The specialization is reflected in the assumptions made about data quality, tree density, crown shape, and the type of non-tree objects present in a scene. In this work, we focus on ITS in urban 3D point clouds captured from below the canopy using TLS, MLS, or PLS. Existing approaches for this setting can be grouped into clustering and region growing approaches, graph-based approaches, and DL approaches. Clustering approaches include a supervoxel-based uphill clustering algorithm proposed by Li et al. (2021) and the approach of Hao et al. (2022), in which the tree canopy is divided into several height layers. Points within each layer are then clustered using the DBSCAN algorithm, and clusters covering multiple tree crowns are split using a symmetry rule. Ning et al. (2022) propose a clustering approach in which tree tops are detected and then a layer-by-layer clustering is performed to delineate tree crowns. Both Wu et al. (2013) and Li et al. (2016) propose voxel-based region growing algorithms that use trunk sections as seeds and then expand them downward and upward. Graph-based approaches can be divided into those based on graph-cut algorithms and those based on path-finding algorithms. The former convert 3D point clouds into weighted graph representations and use graph cut algorithms, such as Normalized cut, to segment the graph into disjoint node sets representing individual trees (Zhong et al., 2017; Chen et al., 2019; Hirt et al., 2021). Hui et al. (2022) propose a path-finding approach, in which 3D point clouds are voxelized, a trunk detection is performed, and the voxels are assigned to the trunk to which they have the shortest path. The application of DL to ITS has been explored recently by Luo et al. (2021), Wang et al. (2023), and Xiang et al. (2023). The proposed DL architectures use voxel or supervoxel representations of 3D point clouds as input and predict offset or embedding vectors for each voxel or supervoxel, which are clustered to obtain the grouping into individual trees.

In our algorithm, we combine two existing techniques for coarse-to-fine ITS, namely the marker-controlled Watershed algorithm and a 3D region growing algorithm. By doing so, we build upon previous work in the following areas.

Marker-Controlled Watershed Segmentation. The marker-controlled Watershed algorithm is an image segmentation technique that can be used for ITS by applying it to a 2D canopy height model. Since it is computationally efficient and resilient to point sparsity, it is commonly used for ITS in large-scale 3D point clouds acquired by ALS, often in conjunction with post-processing steps that further refine the segmentation (Yang et al., 2020; Liu et al., 2024).

Region Growing Approaches. While Wu et al. (2013) and Li et al. (2016) proposed voxel-based region growing algorithms for ITS, we use a custom region growing algorithm that directly processes 3D point clouds. As such, our region growing algorithm is similar to that proposed by Tockner et al. (2022) for 3D point clouds of dense, natural forests.

Coarse-to-Fine Approaches. Li et al. (2023) propose a coarse-to-fine approach for ITS in urban PLS point clouds. They use the DBSCAN algorithm for a coarse clustering of tree points, split clusters containing multiple trees by vertical planes, and refine the segmentation by iteratively applying DBSCAN clustering and k-nearest neighbor classification. Our approach is similar, yet we employ a more computationally efficient algorithm for coarse segmentation and, in addtion, consider data quality as a criterion for determining whether to refine a segmentation.

Approaches Based on Semantic Segmentation. Several previous works perform DL-based semantic segmentation before ITS (Ning et al., 2022) or in combination with DL-based ITS (Xiang et al., 2023; Wang et al., 2023). However, previous DL approaches only distinguish between tree and non-tree points. In contrast, we generate a more detailed segmentation, further distinguishing between tree trunks and crowns, and use this additional information to select markers and seed points for ITS.

3. Coarse-To-Fine Algorithm for Tree Instance Segmentation Based on Semantic Segmentation

We propose an algorithm for coarse-to-fine ITS in urban 3D point clouds. Our algorithm requires a prior semantic segmentation of the 3D point clouds into trunk, crown, and non-tree points. The algorithm combines two existing ITS techniques, the computationally efficient marker-controlled Watershed algorithm and a more accurate 3D region growing algorithm. Our algorithm consists of the following steps (Fig. 1): (1) Tree positions are localized in the 3D point clouds by searching for trunks and crown tops on the basis of semantic segmentation. (2) The marker-controlled Watershed algorithm is used to coarsely delineate tree instances based on a 2D canopy height model, using the tree positions as markers. To remove errors in the Watershed segmentation, we extend the Watershed algorithm with a post-processing step based on a Voronoi segmentation. (3) Using the coarse segmentation, areas with overlapping tree crowns are automatically detected. If data quality is sufficient, the segmentation of these areas is further refined using a 3D region growing algorithm.

3.1 Localization of Tree Positions

The objective of the first step is to identify the 2D positions of individual trees. To ensure robustness against incomplete data, both trunks and crown tops are considered as indicators of tree positions. Trunk and crown top positions corresponding to the same tree are merged in a matching step.

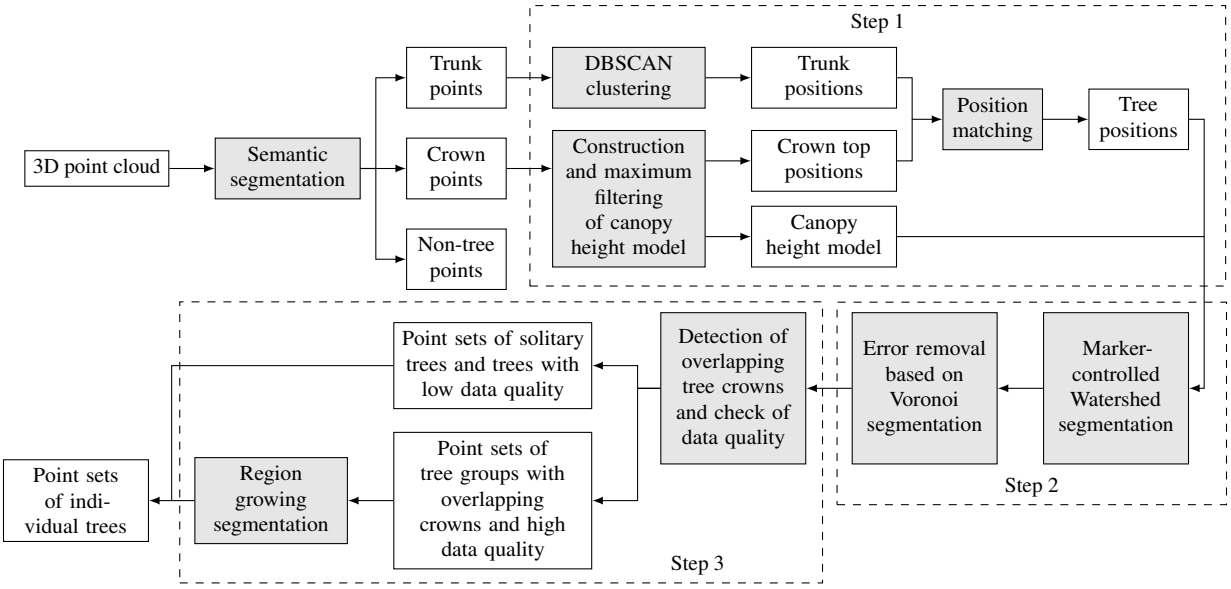


Figure 1. Overview of our algorithm for coarse-to-fine tree instance segmentation. White boxes represent data and gray boxes represent data processing steps. Semantic segmentation is not part of our algorithm, but must be accomplished beforehand.

Localization of Trunk Positions. To obtain the trunk positions, all trunk points identified during semantic segmentation are clustered using the DBSCAN algorithm (Ester et al., 1996). Each resulting cluster is considered to represent a separate trunk, and the trunk position is set to the mean of the X- and Y-coordinates of the points in a cluster. Since the trunks of adjacent trees are usually separated, and we want to avoid oversegmentation in low-density point cloud regions, we set the hyperparameters of the DBSCAN algorithm to values that allow the clusters to grow across larger gaps, namely $\epsilon = 2.5\,\mathrm{m}$ and $MinPoints = 1$. All clusters with less than 100 points are discarded, as such small clusters usually result from semantic segmentation errors.

Localization of Crown Top Positions. To locate the crown top positions, a grid-based 2D canopy height model is constructed using a grid size of $0.5\,\mathrm{m}$ (Fig. 2a). Each grid cell of the canopy height model stores the maximum z-coordinate of all contained crown points. After smoothing the canopy height model with a Gaussian filter, it is searched for local maxima with a search radius of $3.5\,\mathrm{m}$. Local maxima whose height is less than $2.5\,\mathrm{m}$ are discarded because all trees are expected to be higher. To reduce oversegmentation in sparse point cloud regions, local maxima are also discarded if the corresponding grid cell contains less than 100 points. The grid positions of the remaining local maxima are converted to point coordinates and used as crown top positions.

Matching of Trunk and Crown Top Positions. If the horizontal distance between a trunk and a crown top position is less than $5\,\mathrm{m}$, we assume that both positions correspond to the same tree. In this case, we only use the crown position as a marker for the subsequent Watershed segmentation, while trunk positions that cannot be matched to any crown top position are kept as additional markers.

3.2 Coarse Tree Instance Segmentation

Since the accurate delineation of individual trees using region growing algorithms is computationally expensive, we first perform a coarse tree instance segmentation and refine the results only for tree groups with overlapping crowns. For the coarse segmentation, the marker-controlled Watershed algorithm (Kornilov and Safonov, 2018) is applied to the inverse canopy height model, using the tree positions detected in the previous step as markers. As shown in Fig. 2b, applying the Watershed algorithm to the inverse canopy height model results in a coarse, two-dimensional segmentation mask. In some cases, e.g., for subordinate trees that are close to larger trees, the Watershed segmentation of the canopy height model produces incorrect results. Cases of incorrect segmentation can be identified by the fact that only one pixel is assigned to a tree in the segmentation map, or that the pixels assigned to a tree are completely enclosed by pixels belonging to another tree. We implement a post-processing step that automatically detects such cases and replaces the Watershed labels of the affected trees with a Voronoi segmentation (Fig. 2c). In the Voronoi segmentation, each pixel is assigned to the closest tree location. The corrected segmentation mask is then used to determine whether the crowns of adjacent trees touch. If the crown of a tree does not touch the crown of any other tree in the segmentation mask, its segmentation is considered accurate and will not be refined further. Otherwise, the data quality is assessed to determine whether it is sufficient to refine the segmentation through region growing. Specifically, the segmentation of trees with touching crowns is refined if they meet the following criteria: (1) At least one tree trunk point must have been detected during semantic segmentation for a tree. (2) The average distance between a tree's points and their nearest neighbors must not exceed a threshold, which was set to $0.06\,\mathrm{m}$ on the basis of preliminary experiments. If a tree does not meet these criteria, it may be located in a sparse region of a 3D point cloud or be part of a vegetation stand with dense undergrowth. In such cases of low data quality, it is unlikely that the region growing algorithm will improve the coarse segmentation, and the coarse segmentation is retained for that tree.

3.3 Refined Tree Instance Segmentation

For trees that fulfill the aforementioned criteria, the labels from the Watershed segmentation are discarded and their segmentation is refined using a region growing algorithm. Prior to region growing, a set of seed points must be found for each tree to be segmented that belong to that tree with high confidence. Since the trunks of adjacent trees are usually well separated, we use all

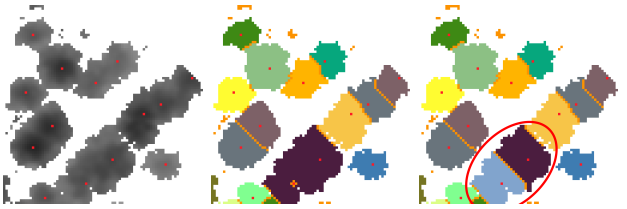
trunk points of a tree as seed points. These trunk points are obtained from the trunk clusters identified in step 1. Using the seed points as initialization, a list of points to be processed is maintained during region growing. For each point to be processed, the predecessor point from which it was reached is tracked, and the distance to and the tree ID of the predecessor point are stored. For seed points, no predecessor points exist, and thus, the distance is initialized to zero, and the tree ID is obtained from the clustering of trunk points in step 1. The points are then processed in ascending order according to their distance to the predecessor point. When a point is processed, it is assigned to the same tree as its predecessor point. Its k-nearest neighbors within a maximum distance of $1\,\mathrm{m}$ are retrieved (k = 27), and those neighbors that are not yet assigned to any tree are added to the list of points to be processed. If neighboring points are already contained in the list of points to be processed, this indicates that they were reached from other points before. In this case, the points are assigned to the closest predecessor point. The region growing continues until the list of points to be processed is empty. Processing the points in order of their distances to the predecessor ensures that the region growing is initially limited to points that are closely connected, and only passes over larger gaps if points cannot be reached in another way. Given that the crowns of neighboring trees are typically separated by gaps that are larger than the distance between points within a tree, this approach allows for the delineation of neighboring tree crowns. However, if tree crowns are strongly overlapping, the region growing of one tree can still extend to neighboring trees. To address this issue, we incorporate information from the coarse segmentation into the region growing process: For each point to be processed, the distance to the crown boundary that was obtained during coarse segmentation is determined. For points outside the coarse crown boundary, the distance to their predecessor points is doubled, thus reducing their processing priority. This ensures that a tree only grows beyond the boundaries identified in the coarse segmentation if the respective tree parts are separated from the neighboring trees by sufficiently large gaps. To determine whether a given point is located outside the coarse crown boundary in an efficient manner, a grid-based 2D signed distance field is generated from the coarse segmentation map of each tree. The grid cells store the 2D distance to the crown boundary, with the distance for grid cells within the crown boundary being multiplied by -1. To ascertain whether a given point is situated within the crown boundary, its coordinates are projected onto the grid, and the sign of the value in the corresponding grid cell of the signed distance field is checked. To promote upward growth, we further scale the z-coordinates of all points by a factor of 0.5 before region growing.

4. Evaluation Methods

4.1 Datasets

We evaluated our algorithm on two custom MLS datasets from Essen, Germany, and Hamburg, Germany, and on a subset of the publicly available TreeML dataset, which is a TLS dataset from Munich, Germany (Yazdi et al., 2024). An overview of these datasets is provided in Table 1.

The MLS datasets from Essen and Hamburg were collected using a Trimble MX8 LiDAR mounted on a vehicle and cover streets with varying tree densities, including both solitary trees and trees with highly overlapping crowns. The Essen dataset was collected during the foliage season, while the Hamburg dataset was acquired in the leafless season, thus capturing branches within the canopy with higher density. The 3D point clouds from both datasets were manually segmented using the CloudCompare software[1] to create ground truth labels for semantic segmentation and tree instance segmentation. For semantic segmentation, the points were divided into the categories ‘low vegetation’, ‘tree trunk’, ‘tree branch’, ‘tree crown’, and ‘other’. Since our ITS algorithm does not distinguish between trunk and branch points, points that were assigned to the ‘tree branch’ category during semantic segmentation were re-assigned to the ‘tree trunk’ category before ITS. To train and evaluate DL models for semantic segmentation, training, validation, and test sets were compiled, including 3D point clouds from Essen and Hamburg in each subset. For the evaluation of the ITS approach, only the 3D point clouds from the validation and test sets were used.



(a) Canopy height model (b) Watershed segmentation (c) Watershed seg. + error removal

Figure 2. Example of intermediate results of the first two steps of our algorithm. Tree positions are shown in red. In (c), an area is outlined in which a segmentation error has been corrected.

Figure 3. Results of the DL semantic segmentation approach for the 3D point cloud HAG1. Segmentation errors are marked in red.

The TreeML dataset was acquired during the leaf-off season using a vehicle-mounted Riegl VZ-400i LiDAR sensor, with the vehicle stopping at each scan position (Yazdi et al., 2024). Compared to the MLS data, the trees are captured with less occlusion and higher point density. The dataset includes ground truth labels for semantic segmentation and tree instance segmentation. The semantic segmentation labels were created using a semi-automatic approach and cover the categories ‘tree’, ‘building’, and ‘other’. In this work, the ‘building’ category was merged with the ‘other’ category. The tree instance segmentation labels were created manually using the CloudCompare software. In this work, a subset of five 3D point clouds containing scenes with dense roadside vegetation was selected for evaluation.

4.2 Semantic Segmentation

Our approach requires as input the results of a previous semantic segmentation. In our evaluation, we considered two scenarios: (1) The semantic segmentation was performed manually, using the ground truth labels for the semantic segmentation as input.

[1] CloudCompare software: `https://cloudcompare.org/`

Since the TreeML dataset does not contain ground truth labels for the categories 'tree trunk' and 'tree crown', this scenario was only tested for the Hamburg and Essen datasets. (2) Semantic segmentation was performed automatically using the DL approach described by Burmeister et al. (2023). Specifically, a KP-FCNN Rigid model (Thomas et al., 2019) was trained on the training set compiled from the Essen and Hamburg datasets, and the model was then used to infer semantic segmentation labels for the TreeML dataset, and for the validation and test sets of the Essen and Hamburg datasets (Fig. 3). The IoU scores of the DL-based semantic segmentation are provided in Table 1.

4.3 Data Preprocessing

To improve computational efficiency, the density of the input point clouds was reduced for both DL-based semantic segmentation and tree instance segmentation. For this purpose, a voxel-based downsampling with a voxel size of $0.03\,\mathrm{m}$ was used. After semantic segmentation and tree instance segmentation, the labels were extrapolated to the original density by assigning each unlabeled point the label of the nearest labeled point. The evaluation metrics were calculated based on the extrapolated labels.

4.4 Evaluation Metrics

To evaluate the ITS accuracy, a matching of the predicted tree instances with the ground truth instances was performed using the method of Kirillov et al. (2019). For this purpose, the pairwise Intersection over Union (IoU) between each predicted instance $\mathcal{P}_i$ and each ground truth instance $\mathcal{G}_j$ is calculated. If the IoU of $\mathcal{P}_i$ and $\mathcal{G}_j$ is strictly greater than 0.5, both instances are matched. After matching, the number of true positive (TP), false positive (FP), and false negative (FN) instances is calculated. Predicted instances that do not match any ground truth instance are counted as FP and ground truth instances that do not match any predicted instance as FN. Based on this, the panoptic quality (PQ) was calculated as a combined indicator of instance detection and instance segmentation quality (Kirillov et al., 2019). In addition to the PQ, we considered separate metrics for instance detection and instance segmentation: To measure the quality of instance detection, we calculated the F_1-score, precision, and recall. To measure the quality of pointwise instance segmentation, we calculated the pointwise IoU, precision, and recall for each matched pair of a predicted and a ground truth instance. The metrics were averaged over all correctly detected tree instances, yielding mIoU, mPrecision, and mRecall.

4.5 Ablation Study

To demonstrate the feasibility of our approach, the following variants of our approach are compared in an ablation study:

W-C: In this variant, the marker-controlled Watershed algorithm is used for ITS. To obtain the markers, a maximum filter is applied to a canopy height model constructed from all tree points. Semantic segmentation is used solely for separating tree and non-tree points, and the region-growing refinement is not used.

W-TC: In this variant, additional information from the semantic segmentation is used to select the markers for the Watershed segmentation. As described in Section 3, the markers are obtained by localizing and matching trunk and crown top positions. As in W-T, no region-growing refinement is applied.

W-V: In this variant, the Voronoi-based error removal step for the Watershed segmentation is added in comparison to W-TC.

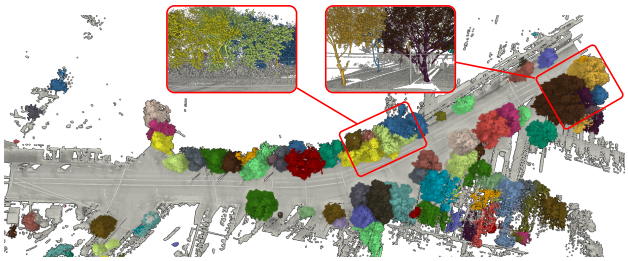


Figure 4. Results of our algorithm for the 3D point cloud EAE5 using DL-based semantic segmentation as input.

W-RG: This variant is the complete algorithm, including all steps described in Section 3.

The ablation study was conducted using DL-based semantic segmentation as input.

5. Results and Discussion

The results of the ablation study are shown in Fig. 5. The full version of our algorithm (W-RG) outperforms the standard marker-controlled Watershed algorithm (W-C) in terms of PQ, with a difference of 0.037, 0.255 and 0.296 in the Essen, Hamburg and TreeML datasets, respectively. Considering the total PQ, all steps of our algorithm contribute to the improvement compared to the baseline: W-TC, which uses trunk positions as additional markers for the Watershed segmentation, outperforms W-C, except for the EAD1 point cloud. For the Hamburg and TreeML datasets, W-TC outperforms W-C by a large margin, demonstrating the advantages of using information derived from semantic segmentation to select markers. The extension of the marker-controlled Watershed algorithm by a Voronoi segmentation-based error removal (W-V), slightly improves the results compared to W-TC in most cases. Overall, the error removal has only little impact on the PQ scores, as the marker-controlled Watershed algorithm already segments most trees with sufficient accuracy. Incorporating the fine segmentation step (W-RG) does not result in an overall improvement on the Essen dataset compared to W-V and even decreases the PQ score for the EAD2 and EAE5 point clouds. This can be attributed to the fact that the Essen dataset was captured during the foliage season, and its 3D point clouds contain areas with strong occlusions and low point density. The data quality criteria proposed by us fail to exclude some of these low-quality areas from region growing, which results in additional segmentation errors. In contrast, for the Hamburg and TreeML datasets, the refinement of the segmentation by the region growing algorithm considerably improves the results, effectively delineating highly overlapping tree crowns (Fig. 6 and Fig. 7).

The metrics of the full version of our algorithm (W-RG) are shown in Table 2. The TreeML (0.839) and Hamburg (0.727) datasets exhibit considerably higher PQ scores compared to the Essen dataset (0.475). These variations can be attributed to the type of scene and the quality of the 3D point clouds: Solitary street trees are reliably detected by our algorithm in all datasets. Accordingly, high PQ scores are achieved for EAD1 (0.767) and EAD2 (0.748) in the Essen dataset, which mainly contain solitary street trees. For the other 3D point clouds, which contain dense tree stands, higher PQ scores are achieved for the Hamburg (0.727 in total) and TreeML (0.839 in total) datasets than for the Essen dataset (0.5 for EAE4 and 0.386 for EAE5). Unlike the Essen data, the Hamburg and TreeML datasets were collected during the leaf-off season. In 3D point clouds recorded

Dataset / 3D Point Cloud	Set	Points	Tree Points	Trees	Complete Trees[a]	IoU			
						Tree	Trunk	Branch	Crown
Essen									
Altendorfer Straße, part 1 (EAD1)	val	7 236 961	488 236	24	21	0.961	0.72	0.395	0.954
Altendorfer Straße, part 2 (EAD2)	val	22 855 907	632 374	10	9	0.947	0.335	0.603	0.978
Altenessener Straße, part 4 (EAE4)	val	19 240 364	5 037 108	56	31	0.958	0.758	0.802	0.946
Altenessener Straße, part 5 (EAE5)	test	19 405 230	4 315 482	136	84	0.909	0.572	0.638	0.905
Hamburg									
Armgartstraße, part 1 (HAG1)	val	25 497 762	4 484 567	28	20	0.962	0.8	0.427	0.942
Armgartstraße, part 2 (HAG2)	test	44 673 629	10 220 439	53	41	0.971	0.777	0.298	0.945
TreeML[b]									
2023-01-09_tum_campus (TTC)	test	48 081 653	14 741 002	65	-	0.992	-	-	-
2023-01-12_57 (T12_57)	test	34 838 587	9 222 208	28	-	0.985	-	-	-
2023-01-12_58 (T12_58)	test	22 520 998	6 080 457	13	-	0.98	-	-	-
2023-01-16_12 (T16_12)	test	31 665 848	10 535 646	35	-	0.992	-	-	-
2023-01-16_43 (T16_43)	test	32 224 288	11 571 550	79	-	0.977	-	-	-

[a] We define complete trees as trees with at least one trunk and one crown point.
[b] Since the TreeML dataset only contains labels for the classes 'tree', 'building', and 'other', no number of complete trees and no IoU scores for the fine-grained vegetation classes are given for this dataset.

Table 1. Overview of the datasets used for evaluation.

Point Cloud	Sem. Seg.	PQ	Instance Detection						Instance Segmentation		
			TP	FP	FN	F_1-score	Prec.	Rec.	mIoU	mPrec.	mRec.
Essen											
Altendorfer Straße, part 1 (EAD1)	GT	0.835	21	2	3	0.894	0.913	0.875	0.934	0.971	0.959
	DL	0.767	18	3	6	0.800	0.857	0.750	0.959	0.981	0.977
Altendorfer Straße, part 2 (EAD2)	GT	0.892	9	1	1	0.900	0.900	0.900	0.991	1.000	0.991
	DL	0.748	8	3	2	0.762	0.727	0.800	0.981	0.995	0.986
Altenessener Straße, part 4 (EAE4)	GT	0.615	41	14	15	0.739	0.745	0.732	0.833	0.901	0.923
	DL	0.500	36	29	20	0.595	0.554	0.643	0.840	0.912	0.918
Altenessener Straße, part 5 (EAE5)	GT	0.479	69	32	67	0.582	0.683	0.507	0.822	0.894	0.916
	DL	0.386	60	50	76	0.488	0.545	0.441	0.791	0.885	0.888
Total	GT	0.576	140	49	86	0.675	0.741	0.619	0.853	0.915	0.929
	DL	0.475	122	85	104	0.564	0.589	0.540	0.842	0.914	0.916
Hamburg											
Armgartstraße, part 1 (HAG1)	GT	0.908	26	0	2	0.963	1.000	0.929	0.943	0.954	0.987
	DL	0.722	24	7	4	0.814	0.774	0.857	0.888	0.922	0.963
Armgartstraße, part 2 (HAG2)	GT	0.823	41	1	12	0.863	0.976	0.774	0.954	0.963	0.990
	DL	0.730	39	7	14	0.788	0.848	0.736	0.927	0.941	0.984
Total	GT	0.854	67	1	14	0.899	0.985	0.827	0.950	0.959	0.989
	DL	0.727	63	14	18	0.797	0.818	0.778	0.912	0.934	0.976
TreeML											
2023-01-09_tum_campus (TTC)	DL	0.785	56	10	9	0.855	0.848	0.862	0.919	0.963	0.952
2023-01-12_57 (T12_57)	DL	0.775	28	14	0	0.800	0.667	1.000	0.968	0.984	0.984
2023-01-12_58 (T12_58)	DL	0.788	13	5	0	0.839	0.722	1.000	0.940	0.992	0.947
2023-01-16_12 (T16_12)	DL	0.925	35	1	0	0.986	0.972	1.000	0.938	0.967	0.968
2023-01-16_43 (T16_43)	DL	0.881	76	10	3	0.921	0.884	0.962	0.956	0.970	0.985
Total	DL	0.839	208	40	12	0.889	0.839	0.945	0.944	0.971	0.971

Table 2. Instance detection and segmentation accuracy of the full version (W-RG) of our algorithm (GT = the semantic segmentation ground truth labels were used as input for the tree instance segmentation, DL = the results of the deep learning method for semantic segmentation were used as input for the tree instance segmentation).

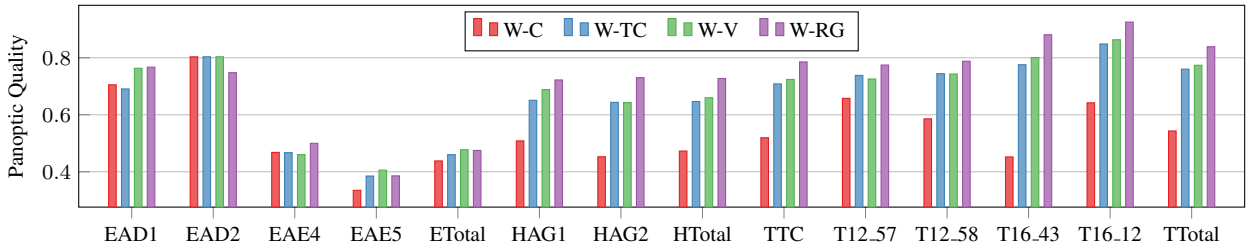


Figure 5. Panoptic quality of the different variants of our algorithm tested in the ablation study (ETotal = Essen, Total; HTotal = Hamburg, Total; TTotal = TreeML, Total; the other abbreviations are the same as in Table 1 and Table 2).

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-4-W5-2024-79-2024 | © Author(s) 2024. CC BY 4.0 License.

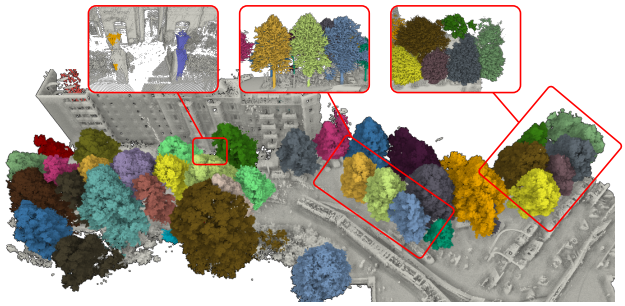


Figure 6. Results of our algorithm for the 3D point cloud HAG2 using DL-based semantic segmentation as input.

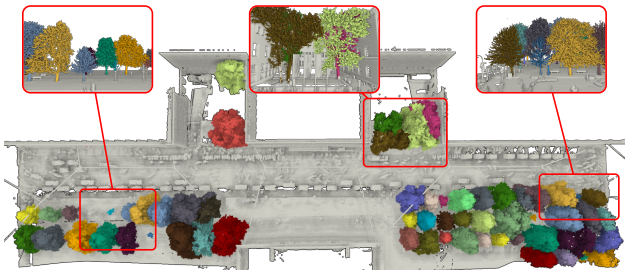


Figure 7. Results of our algorithm for the 3D point cloud TTC using DL-based semantic segmentation as input.

during the leaf-off season, gaps between neighboring crowns and branches within the crowns are more discernible, which is advantageous for the region growing segmentation. Furthermore, our algorithm also benefits from a high point density, as evidenced by the superior PQ scores in the TLS-based TreeML dataset compared to the MLS-based Hamburg dataset. Cases in which our algorithm often produces errors include trees with several codominant trunks, which tend to be oversegmented, and adjacent trees with closely spaced trunks, which tend to be detected as a single tree (middle detail view in Fig. 7). Moreover, trees located at the peripheries of 3D point clouds are prone to segmentation errors, either due to their sparse representation or because they are truncated and thus missed in the first step of our algorithm. The latter occurs especially in HAG2, resulting in several false negatives (right detail view in Fig. 6).

Employing ground truth labels for semantic segmentation (GT) instead of DL-based semantic segmentation (DL) enhances the PQ scores by 0.101 for the Essen dataset and by 0.127 for the Hamburg dataset, indicating that our approach would benefit from more accurate DL methods for semantic segmentation. Precision is more affected by semantic segmentation quality than recall, with precision increasing by 0.151 and 0.167 and recall increasing by 0.08 and 0.049, when substituting DL with GT for the Essen and Hamburg datasets, respectively. Specifically, mislabeling of light poles and building parts as trees during semantic segmentation causes several false positive tree detections (left detail view in Fig. 4 and Fig. 6, respectively).

Table 3 shows the runtime of our algorithm for three exemplary 3D point clouds. The region growing step is notably the most time-consuming. This results in comparably high runtimes for 3D point clouds that contain dense tree stands, necessitating refinement of the coarse segmentation through region growing in most instances ($10.4\,\mathrm{min}$ for HAG2 and $15.2\,\mathrm{min}$ for TTC). For 3D point clouds containing mostly solitary trees, the runtime is considerably shorter ($16.67\,\mathrm{s}$ for EAD1), demonstrating that our approach substantially reduces the execution time by skipping fine segmentation in areas where it is not required.

Processing Step	EAD1	HAG2	TTC
Voxel-based downsampling	1.88	20.54	28.26
Detection of trunk positions	0.46	41.46	42.59
Constr. of canopy height model	0.03	0.22	0.35
Detection of crown top positions	0.02	0.01	0.02
Position matching	0.01	0.01	0.01
Watershed segmentation	0.0	0.0	0.01
Correction of Watershed seg.	0.0	0.0	0.0
Region growing segmentation	13.91	555.81	831.38
Upsampling of labels	0.12	4.25	7.15
Total	16.67	623.99	912.28

Table 3. Runtime of our algorithm (in seconds) for three 3D point clouds from the evaluation datasets on a Dell-XPS-15 notebook.

6. Conclusions and Future Work

In this paper, we presented a coarse-to-fine algorithm for segmenting tree instances in urban 3D point clouds. Initially, our algorithm creates a coarse segmentation through the marker-controlled Watershed algorithm. Subsequently, it identifies regions with overlapping tree crowns and refines their segmentation using a 3D region growing algorithm, provided the data quality is sufficient. Our algorithm achieves high accuracy in segmenting solitary trees and trees with moderately overlapping crowns. For trees with extensive crown overlaps, the accuracy of the results varies, and the best results are obtained for dense TLS point clouds from the leaf-off season. Our algorithm is more accurate than the standard marker-controlled Watershed algorithm, while being computationally more efficient than an approach purely based on 3D region growing. However, the region growing step remains the most computationally expensive part of our algorithm. Given the modular structure of our algorithm, future work could explore alternative approaches for fine segmentation to further improve speed and accuracy. Specifically, incorporating DL-based tree instance segmentation approaches for fine segmentation could be investigated, where our current algorithm could be used in semi-automatically generating training datasets. Our algorithm would also benefit from more accurate DL approaches for the semantic segmentation of vegetation in urban 3D point clouds, as information from semantic segmentation is used for coarse and fine segmentation. Despite the aforementioned possibilities for improvement, our coarse-to-fine algorithm addresses the need for tree instance segmentation methods that are both sufficiently accurate to allow the derivation of dendrometric attributes and sufficiently computationally efficient to scale for 3D point clouds of larger areas. Thus, our approach contributes to the automation of LiDAR-based urban tree inventories across large cityscapes.

Code Availability

A Python implementation of our algorithm is released on Github: `https://github.com/ai4trees/pointtree`.

Acknowledgements

We thank the anonymous reviewers for their valuable feedback. We thank the Department for Geoinformation, Surveying and Cadastre of the City of Essen and the AllTerra Deutschland GmbH for providing mobile mapping data. This work was partially funded by the Federal Ministry of Education and Research, Germany through grant 033L305 (‘TreeDigitalTwins’) and grant 01IS22062 (‘AI research group FFS-AI’).

References

Bennett, L., 2010. Trees and Public Liability – Who Really Decides What Is Reasonably Safe? *Arboric. J.*, 33(3), 141–164.

Burmeister, J.-M., Richter, R., Döllner, J., 2023. Concepts and Techniques for Large-Scale Mapping of Urban Vegetation Using Mobile Mapping Point Clouds and Deep Learning. *Proc. Digit. Landsc. Archit. Conf.*, Wichmann, 451–462.

Chen, Y., Wang, S., Li, J., Ma, L., Wu, R., Luo, Z., Wang, C., 2019. Rapid Urban Roadside Tree Inventory Using a Mobile Laser Scanning System. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, 12(9), 3690–3700.

Comesaña-Cebral, L., Martínez-Sánchez, J., Lorenzo, H., Arias, P., 2021. Individual Tree Segmentation Method Based on Mobile Backpack LiDAR Point Clouds. *Sensors*, 21(18), 6007.

Endreny, T. A., 2018. Strategically Growing the Urban Forest Will Improve our World. *Nat. Commun.*, 9(1), 1160.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proc. 2nd Int. Conf Knowledge Discovery and Data Mining*, AAAI Press, 226–231.

Hao, W., Zuo, Z., Liang, W., 2022. Structure-Based Street Tree Extraction from Mobile Laser Scanning Point Clouds. *Proc. 5th Int. Conf. Image and Graphics Processing*, ACM, 373–379.

Herrero-Huerta, M., Lindenbergh, R., Rodríguez-Gonzálvez, P., 2018. Automatic Tree Parameter Extraction by a Mobile LiDAR System in an Urban Context. *PLOS ONE*, 13(4), e0196004.

Hirt, P.-R., Hoegner, L., Stilla, U., 2021. A Concept for the Segmentation of Individual Urban Trees From Dense MLS Point Clouds. *ISPRS Archives*, XLIII-B2-2021, 171–178.

Hui, Z., Li, Z., Jin, S., Liu, B., Li, D., 2022. Street Tree Extraction and Segmentation from Mobile LiDAR Point Clouds Based on Spatial Geometric Features of Object Primitives. *Forests*, 13(8), 1245.

Kirillov, A., He, K., Girshick, R., Rother, C., Dollár, P., 2019. Panoptic Segmentation. *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 9396–9405.

Kjeldsen-Kragh Keller, J., Konijnendijk, C. C., 2012. Short Communication: A Comparative Analysis of Municipal Urban Tree Inventories of Selected Major Cities in North America and Europe. *Arboric. Urban For.*, 38(1), 24–30.

Kornilov, A. S., Safonov, I. V., 2018. An Overview of Watershed Algorithm Implementations in Open Source Libraries. *J. Imaging*, 4(10), 123.

Li, J., Cheng, X., Wu, Z., Guo, W., 2021. An Over-Segmentation-Based Uphill Clustering Method for Individual Trees Extraction in Urban Street Areas From MLS Data. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, 14, 2206–2221.

Li, L., Li, D., Zhu, H., Li, Y., 2016. A Dual Growing Method for the Automatic Extraction of Individual Trees From Mobile Laser Scanning Data. *ISPRS J. Photogramm. Remote Sens.*, 120, 37–52.

Li, Q., Yan, Y., Li, W., 2023. Coarse-To-Fine Segmentation of Individual Street Trees From Side-View Point Clouds. *Urban For. Urban Green.*, 89, 128097.

Liu, Y., Chen, D., Fu, S., Mathiopoulos, P. T., Sui, M., Na, J., Peethambaran, J., 2024. Segmentation of Individual Tree Points by Combining Marker-Controlled Watershed Segmentation and Spectral Clustering Optimization. *Remote Sens.*, 16(4), 610.

Luo, H., Khoshelham, K., Chen, C., He, H., 2021. Individual Tree Extraction From Urban Mobile Laser Scanning Point Clouds Using Deep Pointwise Direction Embedding. *ISPRS J. Photogramm. Remote Sens.*, 175, 326–339.

Ning, X., Ma, Y., Hou, Y., Lv, Z., Jin, H., Wang, Y., 2022. Semantic Segmentation Guided Coarse-to-Fine Detection of Individual Trees from MLS Point Clouds Based on Treetop Points Extraction and Radius Expansion. *Remote Sens.*, 14(19), 4926.

Säumel, I., Weber, F., Kowarik, I., 2016. Toward Livable and Healthy Urban Streets: Roadside Vegetation Provides Ecosystem Services Where People Live and Move. *Environ. Sci. Policy*, 62, 24–33.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., Guibas, L., 2019. KPConv: Flexible and Deformable Convolution for Point Clouds. *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, IEEE, 6410–6419.

Tockner, A., Gollob, C., Kraßnitzer, R., Ritter, T., Nothdurft, A., 2022. Automatic Tree Crown Segmentation Using Dense Forest Point Clouds From Personal Laser Scanning (PLS). *Int. J. Appl. Earth Obs. Geoinf.*, 114, 103025.

Wang, P., Tang, Y., Liao, Z., Yan, Y., Dai, L., Liu, S., Jiang, T., 2023. Road-Side Individual Tree Segmentation from Urban MLS Point Clouds Using Metric Learning. *Remote Sens.*, 15(8), 1992.

Wu, B., Yu, B., Yue, W., Shu, S., Tan, W., Hu, C., Huang, Y., Wu, J., Liu, H., 2013. A Voxel-Based Method for Automated Identification and Morphological Parameters Estimation of Individual Street Trees from Mobile Laser Scanning Data. *Remote Sens.*, 5(2), 584–611.

Xiang, B., Peters, T., Kontogianni, T., Vetterli, F., Puliti, S., Astrup, R., Schindler, K., 2023. Towards Accurate Instance Segmentation in Large-Scale LiDAR Point Clouds. *ISPRS Annals*, X-1/W1-2023, 605–612.

Yang, J., Kang, Z., Cheng, S., Yang, Z., Akwensi, P. H., 2020. An Individual Tree Segmentation Method Based on Watershed Algorithm and Three-Dimensional Spatial Distribution Analysis From Airborne LiDAR Point Clouds. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, 13, 1055–1067.

Yazdi, H., Shu, Q., Ludwig, F., 2023. A Target-driven Tree Planting and Maintenance Approach for Next Generation Urban Green Infrastructure (UGI). *Proc. Digit. Landsc. Architect. Conf.*, Wichmann, 178–185.

Yazdi, H., Shu, Q., Rötzer, T., Petzold, F., Ludwig, F., 2024. A Multilayered Urban Tree Dataset of Point Clouds, Quantitative Structure and Graph Models. *Sci. Data*, 11(28).

Zhong, L., Cheng, L., Xu, H., Wu, Y., Chen, Y., Li, M., 2017. Segmentation of Individual Trees From TLS and MLS Data. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, 10(2), 774–787.