Automated Canopy Delineation from Airborne LiDAR and Orthophoto in an Indonesian Peatland Restoration Landscape

Biswadeep Sen[1], **Ronald Vernimmen**[2] **and Kuldeep Meel**[1]
[1]School of Computing, National University of Singapore
[2]Data for Sustainability, Axel 4571 AK, The Netherlands
t0921318@u.nus.edu
ronald.vernimmen@dataforsustainability.com
meel@comp.nus.edu.sg

Abstract

Light Detection and Ranging (LiDAR) has been widely applied to characterize the structure of forests as it generates three-dimensional point data with high spatial resolution and accuracy. Canopy delineation, which is commonly derived from the canopy height model can be used in a wide variety of problems in forestry and ecology. While there has been much progress in individual tree crown segmentation, automated delineation of canopy or clusters of trees close to each other from various data sources has not been explored. This study compares different approaches to accurately delineate canopy from a discrete LiDAR point cloud and orthophotos over a rewetted and retired plantation on tropical peat in South Sumatra, Indonesia. We apply three different automated methods to delineate canopy and find that an individual tree crown segmentation-based approach on the LiDAR point cloud achieves the highest intersection over union score of 0.9096 with human-annotated ground-truth labels. The results support our claim that machine learning-based approaches can lead to reliable and cost-effective canopy delineation.

1 Introduction

Indonesian peatlands play an essential role in the carbon cycle of the world, and the local economy of the country [Taufik *et al.*, 2020]. Despite occupying less than 10 per cent of the country's land area, peatlands are responsible for holding the same amount of carbon stock as mineral soils in Indonesia [Minasny *et al.*, 2017]. Many studies have explored the role of peatland transformations in causing significant carbon emissions [Baccini *et al.*, 2017; Wijedasa *et al.*, 2018]. In response to the extensive forest and peatland fires in 2015, Indonesia committed to restoring 2 million hectares of peatland [Republic of Indonesia, 2016; Wijedasa *et al.*, 2017]. Restoration of ecosystem functioning involving rewetting and reforesting of degraded tropical peatlands in Southeast Asia aims to re-establish a self-regulating hydrological system. Such restoration could stop the ongoing release of carbon from peat oxidation and fires and protect biodiversity [Hooijer *et al.*, 2010]. In order to analyse the effectiveness of such efforts, monitoring vegetation development of restored peatlands is essential. One way of achieving that is to assess the canopy.

The term canopy is defined as the combination of all leaves, twigs, and small branches in a stand of vegetation [Parker and others, 1995]. Canopy plays a vital role in the efficient recycling of nutrients in the ecosystem [Prescott, 2002]. Structural elements of the canopy provide essential habitats for wildlife species. The plants in canopies possess great chemical diversity and potential benefit to the field of ethnobotany and show promise for modern medicine [Lowman and Rinker, 2004]. Therefore, delineating canopy is an essential task in ecology and forestry.

Traditionally for canopy segmentation tasks, the area under consideration is manually interpreted by human annotators with very high-resolution geospatial images. This task is very time-consuming and challenging to accomplish [Dechesne *et al.*, 2016]. In a country like Indonesia with a vast peatland extent, automated methods for delineating canopy from remote sensing data can potentially provide a reliable and cost-effective way for monitoring vegetation development of restored peatland areas.

This study is the first of its kind to automatically delineate the canopy in a rewetted and retired plantation on tropical peat to aid monitoring vegetation development. In this study, we automatically delineate canopy by applying both a density-based clustering approach as well as an individual tree crown segmentation-based approach on the LiDAR point cloud and deep learning-based object detection on orthophotos. We perform an extensive empirical evaluation of these methods with the human-annotated ground truth. The best performing method gets an intersection over union score of 0.9096 with respect to the ground truth.

The rest of the paper is structured as follows: Section 2 explains some essential terms related to the paper. Section 3 describes prior work related to canopy delineation. Section 4 describes the data collection and provides details of our approaches. Subsequently, we discuss the evaluation methods in Section 5 and conclude in Section 6.

2 Background

This section provides background of the underlying technology used in this study and the algorithms and libraries used.

2.1 LiDAR and Orthophoto

Light Detection and Ranging (LiDAR) is a remote sensing method that uses light in the form of a pulsed laser to measure ranges or variable distances to the Earth. These light pulses, combined with other data, generate precise, three-dimensional information about the shape of the Earth and its surface characteristics [National Oceanic and Atmospheric Administration, 2022]. An orthophoto is an aerial photograph that is geometrically corrected.

2.2 DBSCAN algorithm

Density-based Spatial Clustering of Applications with Noise (DBSCAN) is a clustering algorithm that is commonly used to detect arbitrarily shaped clusters in a dataset [Ester *et al.*, 1996]. One of the advantages of DBSCAN is that it does not assume the number of clusters in the dataset a priori. In this study, canopies are the clusters. Our dataset also does not know the number of canopies beforehand, making DBSCAN an appropriate algorithm. In this algorithm, all points in the dataset are divided into three groups: core points, reachable points and outliers. There are two main parameters: ε and minPts. A point p is a core point if at least minPts points are within distance ε of it (including p). All points within a distance of ε from p are said to be directly reachable from p. A point q is directly reachable from p if there is a path from p to q such that each pair of adjacent points is at a distance of at most ε from each other. All points which are not reachable from any other point are called outliers or noise points. The full algorithm is shown in Algorithm 1 which makes use of a helper function rangeQ shown in Algorithm 2. During execution, the algorithm iterates through every input data point and checks its number of neighbors. This is done by a call to the rangeQ function on line 5. If its number of neighbors is less than minPts, it is labelled noise shown on line 7, otherwise it is considered a core point and given a new cluster label shown on 10. If it is a core point, we then iterate through all of its neighbors shown on line 12. If this point has already been assigned to a cluster, we ignore it. If the neighbor had previously been labelled noise, it becomes a border point shown on line 14. Otherwise, we label this point and check whether it is a core point or not in line 20. If so, we then iterate through its neighbors, and this continues until all the points are assigned.

2.3 Individual tree crown segmentation from LiDAR

Individual tree crown segmentation is the process of individually delineating trees after spatially locating them and extracting height information [Roussel *et al.*, 2020].

For the individual tree crown segmentation, a commonly used approach is by Dalponte and Coomes (2016) shown in Algorithm 3. This approach consists of several steps. First, a low pass filter smooths over the rasterized canopy height model to reduce the number of local maxima. Then the local maxima are located using a moving window approach. We can choose the moving window to be square shaped or circular. A pixel in such a moving window is designated as a local maximum if its value is greater than all other values in the window and greater than some threshold height above the ground. Each local maximum is a region around which we construct a tree crown. We use the find_trees function from the lidR package to find the local maximum [Popescu and Wynne, 2004]. Next, we extract the height of the four neighbouring points from the canopy height model shown on line 3 and add them to the tree crown region if they satisfy a set of constraints. The neighbor's height should be less than the local maximum, and the horizontal distance from the local maximum should be less than a user-defined threshold. Its height should also be greater than the mean height of all the points added to the crown so far multiplied by another threshold and it should not belong to any other tree crown. In Algorithm 3, the input params denote all the user-defined parameters. The checks are done by the Extend function on line 5. We iteratively keep repeating the procedure for all the neighbours of the cells now included in the tree crown region until no points can be further added. After obtaining the points for each tree crown, we create a convex hull around these points shown on

Algorithm 1 DBSCAN

Input: points, distFunc, ε, minPts
Output: Clusters of points

```
1:  c = 0
2:  for p in points do
3:      if label[p] ≠ None then
4:          continue
5:      N ← rangeQ(points, distFunc, p, ε)
6:      if |N| < minPts then
7:          label[p] = noise
8:          continue
9:      c = c + 1
10:     label[p] = c
11:     S = N.delete(p)
12:     for s in S do
13:         if label[s] == noise then
14:             label[s] = c
15:         if label[s] ≠ None then
16:             continue
17:         else
18:             label[s] = c
19:             N ← rangeQ(points, distFunc, s, ε)
20:             if |N| ≥ minPts then
21:                 S = S ∪ N
22: return label
```

Algorithm 2 rangeQ

Input: points, distFunc, ε, q
Output: All neighbors of q

```
1: N = {}
2: for p in points do
3:     if distFunc(q, p) ≤ ε then
4:         N ∪ {p}
5: return N
```

Algorithm 3 DalponteCoomes

Input: points, treetops, params
Output: Tree crowns as polygons

```
1: treeCrowns = {}
2: for t in treetops do
3:     points = neighbors(t)
4:     for p in points do
5:         if Extend(p, t, params) then
6:             points.append(p)
7:     crown = convexHull(points)
8:     treeCrowns.append(crown)
9: return treeCrowns
```

line 7, and the resulting regions become the final tree crown segmentations [Dalponte and Coomes, 2016].

Another commonly used algorithm is by Silva *et al.* (2016). This algorithm first detects all the treetops marked as local maxima. Then a radius is calculated on the basis of the height of the tree and the diameter of the tree crown. Based on this radius, a local maximum buffer is performed, and the results are cut using Voronoi tessellation to obtain the tree crowns. A third individual tree crown segmentation algorithm is the relative distance algorithm, a point-cloud-based algorithm that uses the relative distance between trees to segment the tree crowns. This algorithm assumes there is always a distance between trees and that the top of the tree crown has a greater distance than the lower section. In this algorithm, the point cloud is normalized at first. Then each point is classified as a target tree or a non-target tree. This process starts at the treetop by including and excluding points based on their relative distances. This process is based on the local maxima and the non-local maxima determined using the tree distance threshold and the minimum distance of the target and non-target trees. The tree distance threshold is determined based on the smallest distance between trees, and the minimum distance of the target tree is determined based on the smallest tree crown diameter [Irlan *et al.*, 2020; Li *et al.*, 2012].

2.4 Tree detection from imagery and DeepForest library

Tree detection from overhead imagery is a considerable problem in ecosystem research, and a lot of scientific and commercial research relies on the accurate delineation of tree crowns. In academic literature, deep learning-based object detection techniques have been used to detect individual tree crowns. One of the main advantages of deep learning over methods that use handcrafted pixel features is that convolutional neural networks (CNNs) directly delineate objects of interest from the training data, improving transferability among projects. Also, neural networks are re-trainable to incorporate the idiosyncrasies of individual datasets, allowing us to refine models with new data without discarding old model [Weinstein *et al.*, 2019]. The DeepForest model [Weinstein *et al.*, 2020] is such a pre-trained model that can be used to detect tree crowns from overhead imagery. This model uses a Retinanet one-stage detector trained using a semi-supervised approach using data from the National Ecological Observatory Network (NEON) site at California's San Joaquin Experimental Range. The way retinanet differs from other object detection frameworks is that it combines detection and classification into a single network, allowing faster training and decreasing the sensitivity to the number of box proposals [Lin *et al.*, 2017]. Here a resnet-50 backbone pre-trained on ImageNet was used as a backbone to retinanet [He *et al.*, 2016].

3 Related work

This section discusses prior work in forest stand segmentation using different algorithms both in Indonesia and other parts of the world.

In Indonesian peatlands, one other study uses tree segmentation-based approaches on a LiDAR point cloud. Irlan *et al.* (2020) applied the same three algorithms as we use in this study Dalponte and Coomes (2016), Silva *et al.* (2016) and Li *et al.* (2012) for tree detection and tree crown diameter estimation in a peat swamp site located in Waringin Timur Regency, Central Kalimantan Province in Indonesia. They found that the Li *et al.* (2012) algorithm performed best by calculating the accuracy of the different methods when compared with field data. The algorithm obtained the highest F-score of 0.63 [Irlan *et al.*, 2020]. Saleh *et al.* (2021) generated a canopy cover estimation model using Landsat 8 OLI and LiDAR in a lowland forest in South Sumatra. Their best estimation obtained an R^2 score of 0.663 and standard deviation of 0.161 between the actual and predicted values.

Outside of Indonesia, Dechesne *et al.* (2016) used a 3-step method for forest stand delineation according to tree species. The method relies on the computation of LiDAR and multispectral features at different levels, used for supervised tree species classification. They tested their approach in a study site located in a mountainous forest in the East of France. The final results obtained a good matching with the segments delineated by human annotators [Dechesne *et al.*, 2016]. In Central European Forests, Sačkov *et al.* (2019) has performed a comparative study between two tree detection algorithms for the estimation of forest stand and ecological variables from airborne LiDAR data.

4 Data and Approaches

This section provides a brief overview of the data collection for this study, and describes the methods we apply to solve the problem.

4.1 Data Collection

The airborne LiDAR and orthophoto were collected over a 2,000 ha retired *Acacia* plantation on peat in South Sumatra, Indonesia, in August 2018, 2.5 years after rewetting of the area. At the time of retirement, *Acacia* trees were partly harvested and partly left standing, resulting in a mixed land cover landscape, allowing monitoring vegetation development under different land cover. We test the automated tree canopy delineation methods described below on a 1 x 1 km (100 ha) area covering both harvested and unharvested *Acacia*. The LiDAR point cloud was used to construct a canopy height model, which denotes the height or residual distance between the ground and the top of objects above the ground.

Algorithm 4 Cluster-Canopy

Input: points, distFunc, ε, minPts
Output: Tree canopies as polygons

```
1: L = {}
2: for p in points do
3:     if p.height ≤ hmax and p.height ≥ hmin then
4:         L.append(p)
5: C ← DBSCAN(L, ε, minPts)
6: canopies = {}
7: for c in C do
8:     k = createHull(c)
9:     canopies.append(k)
10: return canopies
```

4.2 Density-based Clustering approach

We define an approach named Cluster-Canopy for detecting canopy cover from LiDAR point cloud shown in Algorithm 4 using the DBSCAN algorithm on the LiDAR point cloud for points within a predefined canopy minimum and maximum height denoted by hmin and hmax, respectively. Cluster-Canopy first selects the x and y coordinates of all the points whose height lies within that range from the LiDAR point cloud shown on line 3. On line 5, the DBSCAN algorithm is run on the selected points with parameters ε and minimum points = minPts. Once the clusters are determined, the algorithm constructs hulls around them to form the canopies within the specified height range on line 8. We implemented this using the Python package alphashape version 1.3.1 (https://github.com/bellockk/alphashape).

The main challenge is finding the appropriate values of the hyperparameters empirically. After doing a grid search, we got the best results for $\varepsilon = 0.9$ and minPts $= 20$. The best results were evaluated by comparing with the human-annotated ground truth. Lower values of ε resulted in numerous small disjoint clusters, while higher values of ε resulted in the creation of a few big clusters not representing the canopy. In the case of the minPts parameter, also we had similar results. Lower values of minPts created a few large clusters, and higher values created many small clusters, neither of which was useful for properly representing the canopy.

4.3 Individual tree crown segmentation-based approach

We define our approach TC-Canopy shown in Algorithm 5. In this approach, first, we find the treetops in the point cloud using find_trees function available in the lidR package written in R shown on line 1. This function uses a moving window approach to find the local maxima, which are labelled as treetops [Popescu and Wynne, 2004]. Then on line 4 we filter out all the treetops which lie within a predefined canopy maximum and minimum height. Then we use an existing individual tree crown segmentation algorithm on the point cloud and the treetops denoted by itcSegment on line 6. The algorithm returns each segment of the region of interest that contains a tree in the form of a polygon. Then we take the union of the polygons obtained to form the final canopy as shown on line 8. For the individual tree crown segmentation we use three different algorithms: Dalponte and Coomes (2016), Silva *et al.* (2016) and Li *et al.* (2012). Taking the union of the tree crowns often resulted in numerous small holes in the output polygons. While large holes in the output polygons may denote gaps in the canopy, the small holes are not useful for our purpose as they denote tiny gaps between trees. So on line 9 we remove all holes with area less than 5 m^2.

We used the R implementation of the individual tree crown segmentation algorithms from the lidR package version 3.1.4 [Roussel *et al.*, 2020]. To create the union of the tree crowns cascaded_union function was used from the package shapely version 1.8.0 in Python 3.7 [Gillies, 2013]. The small holes were deleted using the deleteholes function in QGIS 3.20.

Algorithm 5 TC-Canopy

Input: points, params
Output: Tree canopies as polygons

```
1: treeTops = findTrees(points)
2: fTtops = {}
3: for t in treeTops do
4:     if t.height ≤ hmax and t.height ≥ hmin then
5:         fTtops.append(t)
6: tCrowns = itcSegment(points, fTtops, params)
7: crownUnion = cascadedUnion(tCrowns)
8: canopy = deleteHoles(crownUnion, 5)
9: return canopy
```

Algorithm 6 OD-Canopy

Input: Orthophoto
Output: Tree canopies as polygons

```
1: boundingBoxes = DeepForest(Orthophoto)
2: return cascadedUnion(boundingBoxes)
```

4.4 Object detection-based approach

We can look at the canopy as just a collection of individual trees. So in this approach, we can use deep learning-based object detection to detect the individual tree crowns from the orthophoto use the results to form the canopy. However, Deep Learning methods, including object detection, require massive amounts of data for training which we lack. Usually, thousands of labelled images are used to train the neural network, but in our case, we only have one image acquired during the time of LiDAR acquisition. So our best option is to use the pre-trained DeepForest model on overhead forest imagery.

Our method OD-Canopy shown in Algorithm 6 goes through the following steps: We use the pre-trained retinanet model DeepForest on the orthophoto and find the bounding boxes denoting the location of trees obtained by object detection on line 1. Once we get all the bounding boxes as an output from the Deepforest model, on line 2 we take the union of all these bounding boxes to form the canopy using the cascaded_union function in the shapely package version 1.8.0 in Python 3.7 [Gillies, 2013].

5 Evaluation setup and Results

We conducted the training and evaluation on high performance machines on compute clusters with the following specifications: Intel Xeon Silver 4108 with Nvidia Titan V and Intel Xeon Silver 4108 with Nvidia Tesla V100.

We tested and validated our methods on a 1x1 km cut out of the LiDAR point cloud, covering both unharvested and harvested Acacia areas. We focused on crowns of trees greater than 15 m, as these trees already have a pronounced tree crown. To evaluate the accuracy of our models, we used human-annotated labels as ground truth. We provided two human annotators with a visualization of the LiDAR point cloud, the orthophoto and a canopy height model at 1 m spatial resolution denoting the heights of the trees in that region. We then asked them to delineate canopy of more than 15 meters in height manually. For the visualization and manual delineation of the canopy, QGIS 3.20 software was used.

The objective of our evaluations was to answer the following questions:

RQ 1 How well does the output of different algorithms match with the human-annotated labels?

RQ 2 Is the total area of tree canopy from the human-annotated labels and the delineation created by the models close to each other?

RQ 3 Is the total number of tree canopies or clusters of trees located close to each other similar in the algorithm outputs and the ground truth?

To answer the first question, we calculate the intersection over union (IoU) between the ground truth and the output of the algorithms. IoU is the most commonly used metric for comparing the similarity between two arbitrary shapes. IoU is invariant to the scale of the problem under consideration. Because of this property, all performance measures used to evaluate segmentation and object detection tasks rely on this metric [Rezatofighi *et al.*, 2019]. Let P denote the canopy as annotated by the human annotator and Q be the region that the algorithm outputs. Let $P_1, P_2, ..., P_n$ be the polygons denoting the delineations of each component of the canopy as annotated by the human annotator. Let $Q_1, Q_2, ..., Q_m$ be the polygons denoting the delineations of each component of the canopy obtained as output to one of the algorithms. The clearly, $P = \bigcup_{i=1}^{n} P_i$ and $Q = \bigcup_{i=1}^{m} Q_i$. We define $|P|$ = the sum of the areas of all the polygons in P. We define $IoU(P, Q)$ as follows:

$$IoU(P, Q) = \frac{|P \cap Q|}{|P \cup Q|}$$

To answer the second question, we calculate the total canopy area of the delineations given by the algorithms, also known as forest cover. We calculate the number of canopies or clusters of trees obtained by the different methods for the third question. The results obtained by the comparison is shown in Table 1. The results obtained from using different individual tree crown segmentation algorithms in TC-Canopy are shown in Table 2.

We find that our methods Cluster-Canopy and TC-Canopy can match with the human-annotated labels with high accuracy. The total area of canopy cover and the number of canopies detected are also relatively close to that of the human-annotated labels. Among these methods, TC-Canopy with Dalponte and Coomes (2016) resulted in the highest scores on all three metrics. However, our method OD-Canopy gets considerably low scores on all of these metrics. The reason may be because the first two methods are based on the LiDAR point cloud, which captures the spatial structure of the region much better than the orthophoto. In the third method, the delineation is based on the overhead imagery. This method can be used in an area of interest where no LiDAR point cloud is available.

Method	IoU	total area (m^2)	num canopies
Groundtruth	1	561517	712
Cluster-Canopy	0.8043	475944	1363
TC-Canopy	0.9096	537808	825
OD-Canopy	0.2822	207212	6232

Table 1: Accuracy for the different canopy delineation methods

Method	IoU	total area (m^2)	num canopies
Groundtruth	1	561517	712
Li	0.8011	589745	2480
Silva	0.8644	526706	867
Dalponte	0.9096	537808	825

Table 2: Accuracy of TC-Canopy applying different algorithms

Figure 1 shows the performance of our best method TC-Canopy with Dalponte and Coomes (2016) on a portion of the region of interest. It can be noticed that the algorithm's outputs are quite similar to the ground-truth label, which labels trees higher than 15 meters.

6 Conclusion

This study applied three different methods to delineate canopy from LiDAR point cloud and orthophoto. While existing methods are manual and labour-intensive considering the scale of restoration, our methods can automatically accomplish the task using remote sensing data, making it efficient and cost-effective if such datasets are available. We demonstrated that the best method performs well under extensive evaluations, providing an affirmative answer to the initial question - Can machine learning-based techniques be used for automated delineation of the canopy?

An interesting area to explore in the future is using machine learning-based methods to monitor and predict vegetation development in restored peat landscapes. We plan to apply the different methods used in this study on other forests in Southeast Asia. We also want to determine the effectiveness of deep learning-based instance segmentation on canopy delineation.

References

[Baccini *et al.*, 2017] Alessandro Baccini, Wayne Walker, Luis Carvalho, Mary Farina, Damien Sulla-Menashe, and RA Houghton. Tropical forests are a net carbon source based on aboveground measurements of gain and loss. *Science*, 358(6360):230–234, 2017.

(a) The Orthophoto of a portion of the region of interest

(b) The human-annotated delineation of that area

(c) Results of TC-Canopy with Dalponte and Coomes (2016)

(d) White line is the ground-truth and black line is TC-Canopy

Figure 1: Comparison of the delineation produced by our best method TC-Canopy with the ground-truth

[Dalponte and Coomes, 2016] Michele Dalponte and David A Coomes. Tree-centric mapping of forest carbon density from airborne laser scanning and hyperspectral data. *Methods in ecology and evolution*, 7(10):1236–1245, 2016.

[Dechesne *et al.*, 2016] Clément Dechesne, Clément Mallet, Arnaud Le Bris, Valérie Gouet, and Alexandre Hervieu. Forest stand segmentation using airborne lidar data and very high resolution multispectral imagery. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 41, 2016.

[Ester *et al.*, 1996] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

[Gillies, 2013] Sean Gillies. The shapely user manual. *URL https://pypi. org/project/Shapely*, 2013.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[Hooijer *et al.*, 2010] Aljosja Hooijer, Susan Page, JG Canadell, Marcel Silvius, Jaap Kwadijk, H Wösten, and Jyrki Jauhiainen. Current and future co 2 emissions from drained peatlands in southeast asia. *Biogeosciences*, 7(5):1505–1514, 2010.

[Irlan *et al.*, 2020] Irlan, Muhammad Buce Saleh, Lilik Budi Prasetyo, Yudi Setiawan, et al. Evaluation of tree detection and segmentation algorithms in peat swamp forest based on lidar point clouds data. *Jurnal Manajemen Hutan Tropika*, 26(2):123–132, 2020.

[Li *et al.*, 2012] Wenkai Li, Qinghua Guo, Marek K Jakubowski, and Maggi Kelly. A new method for segmenting individual trees from the lidar point cloud. *Photogrammetric Engineering & Remote Sensing*, 78(1):75–84, 2012.

[Lin *et al.*, 2017] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[Lowman and Rinker, 2004] Margaret D Lowman and H Bruce Rinker. *Forest canopies*. Elsevier, 2004.

[Minasny *et al.*, 2017] Budiman Minasny, Brendan P Malone, Alex B McBratney, Denis A Angers, Dominique Arrouays, Adam Chambers, Vincent Chaplot, Zueng-Sang Chen, Kun Cheng, Bhabani S Das, et al. Soil carbon 4 per mille. *Geoderma*, 292:59–86, 2017.

[National Oceanic and Atmospheric Administration, 2022] National Oceanic and Atmospheric Administration. What is lidar? *U.S Department of Commerce*, 2022.

[Parker and others, 1995] Geoffrey G Parker et al. Structure and microclimate of forest canopies. *Forest Canopies.*, pages 73–106, 1995.

[Popescu and Wynne, 2004] Sorin C Popescu and Randolph H Wynne. Seeing the trees in the forest. *Photogrammetric Engineering & Remote Sensing*, 70(5):589–604, 2004.

[Prescott, 2002] Cindy E Prescott. The influence of the forest canopy on nutrient cycling. *Tree Physiology*, 22(15-16):1193–1200, 2002.

[Republic of Indonesia, 2016] Republic of Indonesia. First nationally determined contribution, 2016.

[Rezatofighi *et al.*, 2019] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 658–666, 2019.

[Roussel *et al.*, 2020] Jean-Romain Roussel, David Auty, Nicholas C Coops, Piotr Tompalski, Tristan RH Goodbody, Andrew Sánchez Meador, Jean-François Bourdon, Florian de Boissieu, and Alexis Achim. lidr: An r package for analysis of airborne laser scanning (als) data. *Remote Sensing of Environment*, 251:112061, 2020.

[Sačkov *et al.*, 2019] Ivan Sačkov, Ladislav Kulla, and Tomáš Bucha. A comparison of two tree detection methods for estimation of forest stand and ecological variables from airborne lidar data in central european forests. *Remote Sensing*, 11(12):1431, 2019.

[Saleh *et al.*, 2021] Muhammad Buce Saleh, Rosima Wati Dewi, Lilik Budi Prasetyo, and Nitya Ade Santi. Canopy cover estimation in lowland forest in south sumatera, using lidar and landsat 8 oli imagery. *Jurnal Manajemen Hutan Tropika*, 27(1):50–50, 2021.

[Silva *et al.*, 2016] Carlos A Silva, Andrew T Hudak, Lee A Vierling, E Louise Loudermilk, Joseph J O'Brien, J Kevin Hiers, Steve B Jack, Carlos Gonzalez-Benecke, Heezin Lee, Michael J Falkowski, et al. Imputation of individual longleaf pine (pinus palustris mill.) tree attributes from field and lidar data. *Canadian Journal of Remote Sensing*, 42(5):554–573, 2016.

[Taufik *et al.*, 2020] M Taufik, B Minasny, AB McBratney, JC Van Dam, PD Jones, and HAJ Van Lanen. Human-induced changes in indonesian peatlands increase drought severity. *Environmental Research Letters*, 15(8):084013, 2020.

[Weinstein *et al.*, 2019] Ben G Weinstein, Sergio Marconi, Stephanie Bohlman, Alina Zare, and Ethan White. Individual tree-crown detection in rgb imagery using semi-supervised deep learning neural networks. *Remote Sensing*, 11(11):1309, 2019.

[Weinstein *et al.*, 2020] Ben G Weinstein, Sergio Marconi, Mélaine Aubry-Kientz, Gregoire Vincent, Henry Senyondo, and Ethan P White. Deepforest: A python package for rgb deep learning tree crown delineation. *Methods in Ecology and Evolution*, 11(12):1743–1751, 2020.

[Wijedasa *et al.*, 2017] Lahiru S Wijedasa, Jyrki Jauhiainen, Mari Könönen, Maija Lampela, Harri Vasander, Marie-Claire Leblanc, Stephanie Evers, Thomas EL Smith, Catherine M Yule, Helena Varkkey, et al. Denial of long-term issues with agriculture on tropical peatlands will have devastating consequences. *Global change biology*, 23(3):977–982, 2017.

[Wijedasa *et al.*, 2018] Lahiru S Wijedasa, Sean Sloan, Susan E Page, Gopalasamy R Clements, Massimo Lupascu, and Theodore A Evans. Carbon emissions from south-east asian peatlands will increase despite emission-reduction schemes. *Global Change Biology*, 24(10):4598–4613, 2018.