Focusing on Object Extremities for Tree Instance Segmentation in Forest Environments

Robin Condat, Pascal Vasseur, Guillaume Allibert

Focusing on Object Extremities for Tree Instance Segmentation in Forest Environments

Robin Condat[1], Pascal Vasseur[2] and Guillaume Allibert[3]

***Abstract*—As part of the development of many robotic systems for the forestry sector, forest scene understanding requires the use of computer vision algorithms. However, this dense and unstructured environment is complex and puts conventional detection approaches to the test. In the case of tree instance segmentation, the presence of closely spaced or even intertwined trees, their highly variable shapes, and complex masks due to their branches and leaves are just some of the challenges to be overcome. For this, specific learning of tree boundaries is required to better distinguish one from another. In this paper, we propose ConvexMask, a convolutional neural network for real-time instance segmentation. ConvexMask opts for a label representation approach with a convex exterior polygon, defined by tree extremities, and a binary mask to handle the detail and occlusions that the label may contain. Experiments conducted on the SynthTree43k dataset show that ConvexMask distinguishes tree extremities better than state-of-the-art networks, resulting in better-quality masks. The code is available at https://github.com/rcondat/convexmask**

***Index Terms*—Deep Learning for Visual Perception, Robotics and Automation in Agriculture and Forestry**

I. Introduction

COMPUTER vision algorithms are essential for forest scene understanding, which is a key step in the development of robotic systems for the forestry sector. They enable the extraction of important information for many tasks in this field, such as mapping [1], navigation [2], species classification [3], tree felling [4], log transportation [5] or forest fire detection [6]. For most of these applications, tree detection is mandatory, making it a major task. As a result, a great deal of work has been carried out on the subject in order to achieve accurate and reliable tree detection in real-time. Even so, the forest is a complex, dense, and unstructured environment. Conventional detection approaches are therefore challenged in this new context, requiring new approaches.

For many applications, tree detection is limited to trunk detection only. This task is simpler because the trunk itself is easier to identify than the whole tree. However, in the context of forest navigation and mapping, it is necessary to detect trees in their entirety to better situate them in a 3D environment and identify navigable areas. As the environment is very dense, precise identification of tree contours is crucial. Instance segmentation is then a computer vision task adapted to meet these constraints. To do this accurately and in real-time, deep learning algorithms like Convolutional Neural Networks (CNN) are required.

Despite the very high accuracies achieved by deep learning algorithms on reference benchmarks, improvements are still needed for the instance segmentation of whole trees in forest scenes. Firstly, in our mapping context, these algorithms need to run on embedded systems, requiring lightweight neural networks. Secondly, trees are generally composed of branches and leaves, making them objects with a complex mask, as shown in Fig 1. Finally, in a forest environment, trees are often very close together or even superimposed on each other, complicating the identification of their contours.

To overcome these problems, our approach focuses on the label representation of object ground truth. Each CNN has its own label representation l'_{GT} for its training, produced from the same ground truth l_{GT} of the training dataset. For instance segmentation, the original labels l_{GT} are in the form of a set of binary masks for each object. The vast majority of instance segmentation CNNs encode these ground truths in the form of a bounding box combined with a compressed binary matrix for each object ($l'_{GT} = (B_{GT}, M_{GT})$), as illustrated in Fig. 1b. These networks first detect objects by predicting their bounding box coordinates and then segment the selected regions to produce the final masks. Consequently, it is essential to have good bounding box prediction to ensure proper downstream object segmentation. However, due to the high density of the forest environment, several trees could have almost identical bounding boxes. It is then difficult to differentiate them, resulting in missed detections.

An alternative approach focuses on object contours by representing masks as polygons ($l'_{GT} = (P_{GT})$), as shown in Fig. 1c. Trees are no longer characterized by 4 coordinates but by a set of points. One of the main advantages of using polygons is that each tree is easily distinguishable from another. Nevertheless, in our context, trees have ground truth masks with many occlusions or are divided into several parts, like the red tree in Fig. 1c. Therefore, representing ground truth with a unique polygon means that annotations are sometimes too coarse. In addition, the encoding parameters are difficult to define since the polygon must have enough vertices to faithfully represent the ground truth without having too many

Manuscript received: October, 5, 2023; Revised December, 13, 2023; Accepted April, 7, 2024.

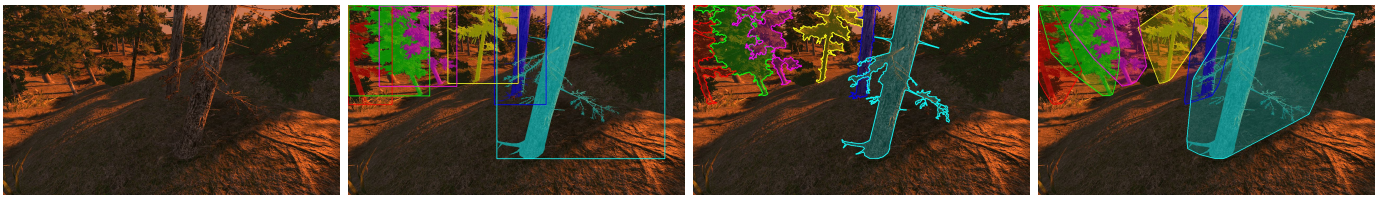
This paper was recommended for publication by Editor H. Moon upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the French National Research Agency through the ANR project CLARA ANR-18-CE33-0004-02 and was performed using the computing resources of CRIANN (Normandy, France).

[1]Robin Condat is with Université de Picardie Jules Verne, MIS, France, also with Université de Rouen Normandie, LITIS, France (e-mail: robin.condat@u-picardie.fr)

[2]Pascal Vasseur is with Université de Picardie Jules Verne, MIS, France (e-mail: pascal.vasseur@u-picardie.fr)

[2]Guillaume Allibert is with Université Côte d'Azur, CNRS, I3S, France (e-mail: guillaume.allibert@etu.univ-cotedazur.fr)

Digital Object Identifier (DOI): see top of this page.



(a) (b) (c) (d)

Fig. 1. Different representations of instance segmentation labels: (a) Original Image (b) Bounding box + binary mask representation (c) Polygon representation (d) Convex exterior polygon + binary mask representation.

so that the polygon is not too complex.

To better characterize the tree's extremities, our work has focused on using a convex polygon encompassing the ground truth mask, as illustrated in Fig. 1d. This convex polygon comprises far fewer vertices while describing the object's location better than bounding boxes. However, it cannot represent ground truth independently and must be combined with a compressed binary mask to manage occlusions and tree features better. In this way, we design ConvexMask, a CNN for real-time instance segmentation, representing labels with an exterior convex polygon and a binary matrix ($l'_{GT} = (C_{GT}, M_{GT})$). ConvexMask detects object extremities upstream with the prediction of a convex polygon, then segments within it to obtain a final mask. With this label representation, we propose Circular Centerness, a sample weighting function adapted to objects with a particular shape, as well as NMS Convex, a post-processing algorithm that takes advantage of convex polygons to eliminate duplicate detections, while better handling highly overlapped objects. Experiments carried out on the SynthTree43k dataset [4] demonstrate the effectiveness of our method, which achieves better detection and segmentation accuracies than state-of-the-art CNNs. Lastly, we qualitatively test ConvexMask trained on synthetic data for prediction in real forest environments.

II. Related Work

Tree detection via computer vision is a subject of study that has had a resurgence of interest in recent years. Numerous studies have proposed tree detection solutions for a variety of applications. These range from species classification [3], [7] to log transport [5], forest mapping [1] as well as fuel detection for preventing forest fires [8], [9], [10]. As a result, the proposed methods focus on certain parts of the tree to be detected, thus extracting the information useful for their application. Where Wang et al. [11] and Fortin et al. [5] will focus on detecting only the trunk of the tree, Russell [8] will identify tree tops in aerial views. Viewpoints also differ according to the robots used for application: drones [12] or Unmanned Ground Vehicles (UGV) [13], [14], to name but a few. The diversity of approaches is also reflected in the sensors used. Wang [7] uses point clouds from Terrestrial Laser Scanning (TLS) for tree isolation and hardwood classification. Itakura and Hosoi [13] perform 3D tree detection from spherical 360° images. In [9], Mendes et al. use RGB images for forest vegetation detection and classification. Finally, tree detection takes a variety of forms. These include bounding box detection [9], 2D segmentation on RGB images [13] or 3D segmentation on point clouds [15], as well as approaches combining 2D detection and 3D reconstruction [11]. Of all these approaches, however, we note that the vast majority do not detect the tree in its entirety. The main reason for this is that the nature of the applications for which these methods are intended does not require them to detect the whole tree. Furthermore, compared with detecting the tree trunk, the task is much more complex.

To be able to detect trees in their entirety, the use of forest scene datasets is necessary. Thanks to the upsurge in forest analysis, numerous datasets have been created, proposing data in various modalities (RGB image, LiDAR, thermal images, stereovision, GPS, etc.) captured from different viewpoints. Grondin et al. [4] introduced SynthTree43k, a synthetic dataset for tree trunk instance segmentation, built from simulated forest environments under various seasons. The dataset comprises 43k samples, with an RGB image and depth map generated, and 162k trees of 17 different species are annotated. SynthTree43k also provides whole-tree labels, although no experiments with these labels have been carried out to date. da Silva et al. [16] propose ForTrunkDet, a dataset of 2029 RGB images and 866 thermal images acquired in three different Portuguese forests, and bounding boxes for tree trunk detection. The FinnWoodlands dataset [17] comprises 5170 samples consisting of stereo RGB images, sparse depth maps, and LiDAR point clouds of Finnish forest scenes. 300 samples comprising 4226 tree trunks are annotated for panoptic segmentation. To the best of our knowledge, there are no real datasets with labeled whole trees for detection or segmentation. This is due to the nature of the targeted application, but also to the complexity of labeling the whole tree (branches, leaves).

III. ConvexMask

In this section, we introduce ConvexMask implementation. We first introduce its overall architecture. Then, we detail the encoding process for converting tree masks into labels for ConvexMask. Next, we present the loss functions used to train the network. Finally, we introduce NMS Convex, a specialized post-processing algorithm for ConvexMask.

A. Architecture

ConvexMask is an anchor-free network based on FCOS [18] architecture and decomposing instance segmentation into two

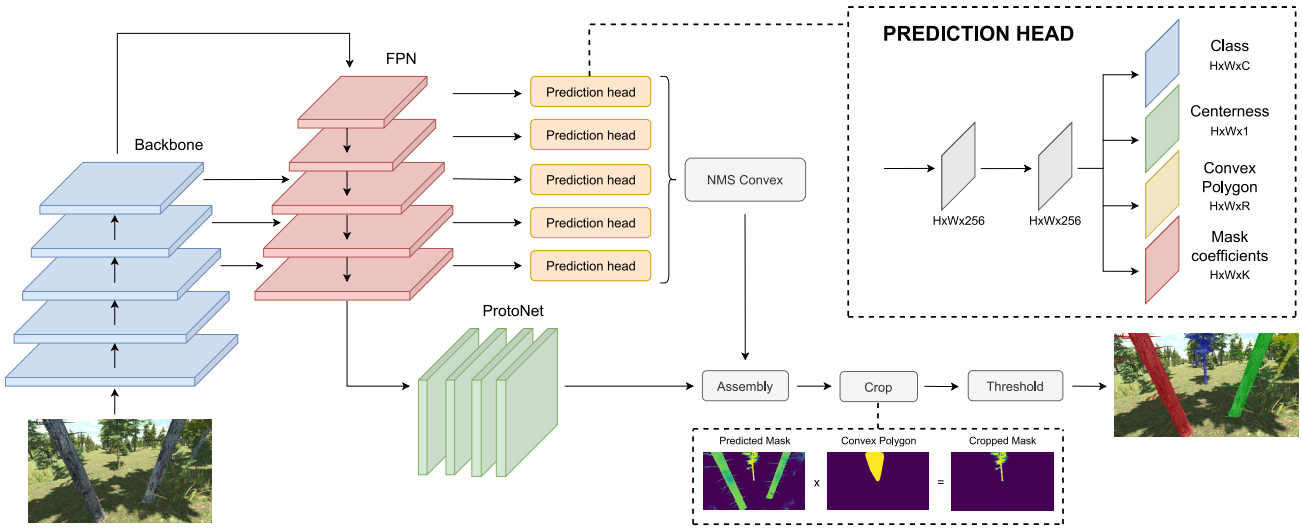


Fig. 2. ConvexMask architecture. H and W are, respectively, the height and width of feature maps, C is the number of classes, R is the number of rays for a convex exterior polygon, and K is the number of prototypes.

parallel sub-tasks. Its architecture, shown in Fig. 2, consists of a backbone, linked to a Feature Pyramid Network (FPN), to produce multi-scale feature maps. Then, image segmentation is performed via a protonet, predicting k prototype masks independently of instances, following Yolact [19] segmentation strategy. In parallel, tree detection is handled by a prediction head, which produces convex polygons in polar format. It is composed of 4 branches to predict for each instance its class, its centerness, its convex exterior polygon, and k coefficients associated with the prototypes for mask assembling.

B. ConvexMask segmentation

In this section, we detail the ConvexMask segmentation process, based on the polar representation introduced with PolarMask [20]. This representation defines a polygon from a center (x_c,y_c) and N rays with a fixed angle interval $\Delta\theta$. As a result, the convex polygon prediction is formulated as instance center classification and dense distance regression in a polar coordinate.

1) Convex Exterior Polygon: Starting with a mask instance, the first step is to define the convex exterior polygon. Edge detection is applied to the mask using the algorithm [21], producing one or more highly detailed contours. From these contours, the Sklansky algorithm [22] is applied to obtain a convex hull that encompasses all previous contours. This convex hull has the advantage of being less detailed, where its vertices are instance extremities, making it an easier polygon to learn. Moreover, this convex hull is guaranteed to contain our entire instance mask. In our context, each convex polygon contains a tree, as shown in Fig. 3b, which can be split into several parts or partially outside the image. It is important to specify that this convex contour does not define the mask instance but a zone that will then be segmented.

2) Instance center: As mentioned earlier, the convex polygon of the instance is defined by a center (x_c,y_c). There are several ways to define this center, such as the bounding box center or mass center. We aim to have a center such that the distances between it and the vertices of the convex polygon vary as little as possible. Therefore, we chose the mass center, shown in Fig. 3b, as adopted by PolarMask [20]. One of the advantages of using convex polygons is that we are sure that their mass center lies within them.

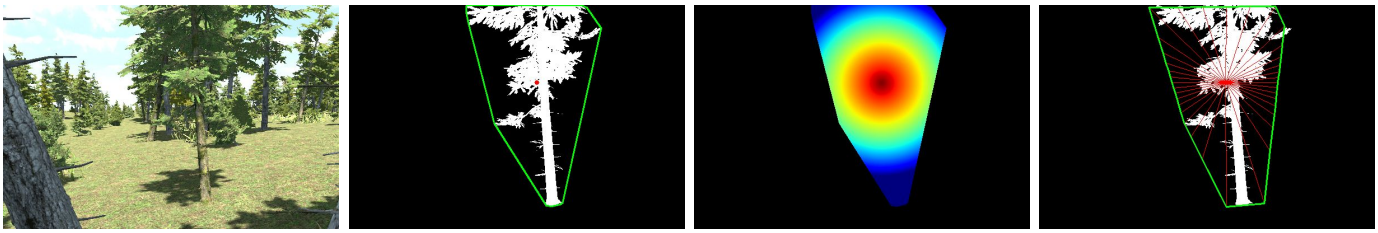
3) Instance samples: For any location (x,y) included in a ground truth convex polygon, we are able to calculate the latter in polar format. Therefore, location is considered as a positive sample if it falls into any ground-truth convex polygon. Otherwise, it is a negative sample. This makes it possible to have several positive samples for a single instance, thus preventing small objects from having no positive sample at all and being ignored.

4) Circular Centerness: The principle of centerness [18] is to weigh the samples during the learning process in order to focus the network on the most relevant samples. For polygon prediction, PolarMask proposed Polar Centerness, calculated according to polygon rays in polar format. Given a set $\{d_1, d_2, ..., d_n\}$ for the length of n rays of one instance, Polar Centerness is defined as follows:

$$Polar\ Centerness = \sqrt{\frac{min(\{d_1, d_2, ..., d_n\})}{max(\{d_1, d_2, ..., d_n\})}} \quad (1)$$

With Polar Centerness, the closer the rays d_{min} and d_{max}, the higher the sample centerness. However, this definition has a major drawback: if the object polygon is thin (i.e. tree trunk), each associated sample will have a considerable difference between its d_{min} and d_{max}, resulting in very low centerness scores. For this reason, we believe that centerness should not be defined according to the polygon shape.

To remedy this problem, we propose Circular Centerness, which aims to give a location a centerness score according to its distance from the ground truth's mass center C. For each sample P, its centerness is defined as follows:



(a) (b) (c) (d)

Fig. 3. Representation of different stages in the label encoding process of one tree : (a) Original Image (b) Ground Truth binary mask with its convex exterior polygon and mass center (c) Instance centerness with scores between 1 (red) and 0 (blue) (d) Ground Truth binary mask with its encoded convex polygon with $N = 36$ rays.

$$Circular\ Centerness = \sqrt{1 - \frac{d_{P,C}}{d_{max}}} \tag{2}$$

where $d_{P,C}$ is the euclidean distance between sample P and center C, and d_{max} the maximum euclidean distance between center C and each polygon vertex. With this definition, the closer the sample P is to the center of the instance C, the higher the centerness. Fig. 3c shows the centerness of each point included in the convex polygon of our example tree. Negative samples have a default centerness of zero. The advantage of Circular Centerness is a simpler definition, independent of polygon shape, better suited to objects with particular shapes like thin trees.

5) Distance regression: Given a positive sample (x, y) and the convex polygon of an instance, the N rays are calculated as follows:

- The vector P of size N is initialized to zero.
- For each point p_i of the convex polygon, their distance d_i and angle α_i from the sample center are calculated.
- Each distance d_i is assigned to the corresponding angle of the polygon vector closest to α_i. If two distances are assigned to the same angle, we choose the greatest.
- For all angles of the polygon vector with an unassigned distance, we calculate their corresponding distance via interpolation with the distances to the nearest angles.
- Polygon distances that are zero, are set to a minimum value of 1e-6.

The construction of this convex polygon in polar format with $N = 36$ rays is shown in Fig. 3d. Depending on the number of rays N defined for the polygon, encoding may produce a polygon that does not entirely encompass the instance mask. If the N value is too low, some extremities of the convex polygon associated with an angle of the polygon vector will be offset by several degrees, creating a less precise polygon. This phenomenon has been taken into account when assembling the mask.

6) Mask assembling: For each location (x,y), during inference, the network predicts its classification, centerness, N rays, and k coefficients associated with the prototypes. Final confidence scores are obtained by multiplying centerness and classification. For each FPN level, the $1k$ locations with the highest confidence scores are retained, and a threshold is applied to select only the top predictions. Convex polygons of these top predictions in the Cartesian plane are constructed via the (x,y) location and the N predicted rays. Next, a post-processing algorithm is applied to eliminate duplicates and produce our final predictions.

For each of these final predictions, the corresponding k coefficients are used to create a binary mask via linear combination with the generated prototypes. This mask is finally cropped with the predicted convex polygons, producing final instance masks. Unlike the bounding box, the convex polygon has a smaller area while still containing the instance mask, reducing segmentation errors due to a nearby similar object. To take into account the accuracy error of the convex polygon due to encoding, convex polygon rays are enlarged by 10% to ensure that they fully encompass our instance.

C. Losses

The loss function used for ConvexMask training is defined as follows:

$$L = \lambda_{cls} L_{cls} + \lambda_{poly} L_{poly} + \lambda_{mask} L_{mask} + \lambda_{cent} L_{cent} \tag{3}$$

We use focal loss [23] for the classification loss L_{cls}, polar IoU loss [20] for the polygon loss L_{poly} (i.e. the loss to regulate the learning of the convex exterior polygon), and the binary cross entropy for the mask loss L_{mask}, calculated in the same way as Yolact [19].

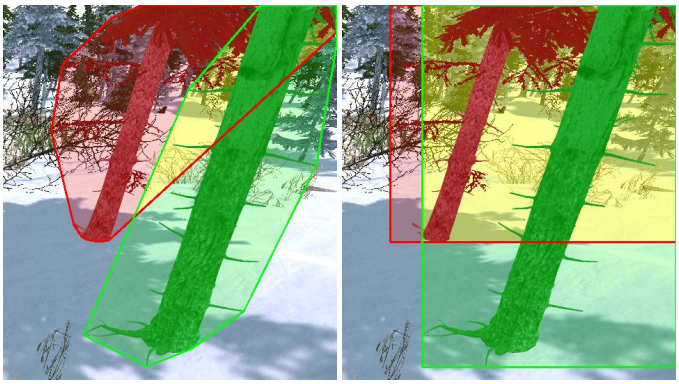
For centerness learning, the loss function often used is binary cross entropy [20], [18], defined as follows:

$$BCE(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} y_i . log(\hat{y}_i) + (1 - y_i) . log(1 - \hat{y}_i) \tag{4}$$

where y_i and $\hat{y}_i$ are the centernesses of ground truth and prediction respectively. However, this loss function is initially designed for binary labels, whereas centerness is a floating score ranging from 0 to 1. Consequently, binary cross entropy does not tend towards zero in this context. To remedy this problem, we defined our centerness loss as follows:

$$L_{cent}(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} log(1 - |y_i - \hat{y}_i|) \tag{5}$$

In this way, the loss function is zero when the prediction is equal to the ground truth. To balance our losses functions, we have set empirically λ_{cls}, λ_{mask}, λ_{cent} to 1 and λ_{poly} to 2.



(a) (b)

Fig. 4. Two close trees and the intersection (in yellow) of their bounding boxes (a) and convex polygon (b).

D. NMS Convex

At the output of the prediction head, we perform Non Maximum Suppression (NMS) to remove duplicate detections, like many instance segmentation networks. The NMS criterion for deleting duplicate detections is the intersection of the union (IoU) of bounding boxes, which is quick and easy to calculate. However, there may be objects with different masks but very similar bounding boxes. NMS then risks deleting one of them with this criterion. Fig. 4a shows an example of two different trees whose bounding boxes have a high IoU (60.5 %). Variants of NMS also exist, calculating the IoU of masks. On the other hand, this operation is more energy-intensive, greatly slowing down inference speed.

With convex polygon prediction, we propose NMS Convex, which is a variant of NMS with the IoU of convex polygons as the deletion criterion. Since convex polygons better describe the instance's locations than bounding boxes, the intersection of two instances of convex polygons is closer to the real instance intersection. Compared to classic polygons, the calculation of the intersection of two convex polygons is easier thanks to one property of convex polygons: the intersection of two convex polygons also forms a convex polygon. Fig. 4b. repeats the previous example, displaying this time the IoU of their convex polygons, which is much lower (16.6 %). The IoU of convex polygons is calculated in polar format, so there is no loss of precision whatever the size of the polygons. However, the process remains slower than for bounding boxes. To reduce this gap, the IoU of bounding boxes is calculated upstream, so that only the IoU of convex polygons whose bounding boxes have an IoU greater than 10 % is considered.

IV. Experiments

We present segmentation instance results on the SynthTree43K [4] benchmark, with whole tree labels. The dataset contains 40k synthetic images for training, 1k images for validation and 2k images for testing, with a total of more than 162k annotated trees. Each instance has the unique label *tree*, and its mask includes the entire tree, taking into account occlusions with other instances or the background.

TABLE I
Impact of Convex Polygon Number of Rays on ConvexMask Performances. The Best Scores are Shown in Bold.

RAYS	AP^B	AP^B_{50}	AP^M	AP^M_{50}
18	61.2	90.2	35.8	74.0
36	65.7	90.6	36.7	75.0
72	**66.4**	90.8	**37.2**	**75.5**
120	66.2	**90.9**	36.9	74.9

TABLE II
Impact of Image Size on ConvexMask Performances and Speed. The Best Scores are Shown in Bold.

IMAGE SIZE	AP^B	AP^B_{50}	AP^M	AP^M_{50}	FPS
640×360	61.2	89.2	28.8	69.3	**36.5**
960×540	64.8	**90.8**	34.4	74.1	34.2
1280×720	**65.7**	90.6	**36.7**	**75.0**	31.6

TABLE III
Impact of Centerness Definition on ConvexMask Performances. The Best Scores are Shown in Bold.

CENTERNESS	VAL SET		THIN SET	
	AP^B	AP^M	AP^B	AP^M
Polar	**66.0**	**37.1**	59.9	40.5
Circular	65.7	36.7	**63.3**	**43.0**

TABLE IV
Impact of Post-Processing Algorithm on ConvexMask Performances and Speed. The Best Scores are Shown in Bold.

NMS	NORMAL		OVERLAP		FPS
	AP^B	AP^M	AP^B	AP^M	
Original	65.1	**36.9**	46.4	12.6	**34.1**
Convex	**65.7**	36.7	**50.8**	**13.3**	31.6

Our models include a ResNet50 backbone [24] (or ResNet101 if specified), pre-trained on ImageNet. We train ConvexMask on a single A100 GPU. Stochastic Gradient Descent (SGD) is used, with an initial learning rate of $1e^{-2}$ and a mini-batch of 8 images. Following the $3x$ learning strategy [25], ConvexMask is trained for 180k iterations, and its learning rate is reduced by a factor of 10 at iterations 135k and 165k. Weight decay and momentum are set to $1e^{-4}$ and 0.9 respectively. Input images retain their original size (1280x720), and horizontal flipping is the only method for data augmentation. We report ablations on the SynthTree43k validation set (1k images). The metrics used to characterize network performance are the standard COCO [26] benchmark metrics, namely the mean Average Precision (AP) and the mean Average Precision with an IoU threshold of respectively 50 % and 75 % (AP_{50}, AP_{75}). These metrics are applied for bounding box prediction (B) and mask prediction (M). It should be noted that ConvexMask bounding boxes are calculated by retrieving the minimum and maximum coordinates of the predicted mask, in length and width.

A. Ablation study

1) Number of rays: As explained in [20], the number of rays plays an important role in polygon prediction. Too few

TABLE V
COMPARISON BETWEEN STATE-OF-THE-ART ALGORITHMS ON SYNTHTREE43K TEST SET. THE BEST SCORES ARE SHOWN IN BOLD.

BACKBONE	MODEL	AP^B	AP^B_{50}	AP^B_{75}	AP^M	AP^M_{50}	AP^M_{75}	FPS
Resnet50	Mask RCNN	64.9	89.4	72.9	28.7	70.0	16.1	28.7
	YOLACT	52.1	78.7	58.7	26.9	57.5	21.4	**35.0**
	PolarMask	42.3	84.1	37.1	8.9	38.5	1.2	22.8
	SOLO	41.2	81.2	36.8	**39.2**	**80.8**	31.5	20.8
	ConvexMask	**66.8**	**90.8**	**75.2**	37.3	75.2	**32.9**	31.6
Resnet101	Mask RCNN	65.2	88.6	73.2	29.1	70.0	16.4	22.5
	YOLACT	53.6	79.4	60.8	28.7	60.5	23.6	**26.0**
	PolarMask	44.0	84.9	40.9	9.3	40.3	1.3	19.1
	SOLO	40.5	80.4	36.1	**38.7**	**80.5**	30.4	17.5
	ConvexMask	**68.4**	**91.4**	**77.1**	38.5	76.7	**34.7**	24.8

rays result in less accurate predictions, while too many saturate performance. PolarMask experiments showed that 36 rays were the most appropriate choice. In our context, the convex polygon designates a segmentation zone, defined with fewer vertices, theoretically requiring less precision.

Table I shows that the number of rays has an impact on ConvexMask precision (AP^B and AP^M metrics). The best performance in terms of accuracy is achieved with 72 rays, separated by a 5-degree angle. The impact of the number of rays on ConvexMask speed is negligible.

2) Speed vs Accuracy: The choice of input image size inevitably impacts network performance and inference speed. Table II shows the speed and accuracy trade-off as a function of input image sizes. The observed results are quite logical, but it is interesting to note that the performance gain between 960×540 and 1280×720 sizes only affects the accuracy of predictions, and not the number of trees detected. Frame per Second (FPS) is indicated on an A100 GPU and corresponds to total inference time, including post-processing. This shows that ConvexMask can be run in real time even on large images. It is important to note, however, that the hardware performance used to measure inference speed is far from the reality of embedded system operation. These measurements are only indicative for comparison between networks.

3) Polar Centerness vs Circular Centerness: As explained in section III-B4, Circular Centerness has been specially designed to better support thin objects by defining their weight independently of the polygon shape. We then compare our proposal with Polar Centerness on the validation set (VAL SET), but also on a subset of this set including only thin trees (THIN SET). Table III shows ConvexMask results according to the centerness used (Polar Centerness or Circular Centerness) on these two sets.

On the original validation set, both methods are equally accurate in terms of bounding box and mask precision. Circular Centerness, on the other hand, stands out on the thin tree subset, with significant performance gains (+3.4% in AP^B and +2.5% in AP^M). This is due to the definition of our proposal, giving higher centerness scores on the positive samples associated with thin trees, thus preventing these proposals from being ignored in post-processing.

4) NMS vs NMS Convex: NMS Convex aims to avoid the deletion of highly entangled trees in post-processing by using the IoU of convex polygons as a deletion criterion, as explained in section III-D. We compare our algorithm with NMS on the validation set (VAL SET) and on a subset of this set comprising only trees with strong overlap with other ground truth trees (OVERLAP SET). The performance and speed of ConvexMask as a function of the post-processing algorithm applied are shown in table IV.

On the validation set, the performance of the two algorithms is equivalent. However, there is a significant improvement in bounding box accuracy on the OVERLAP SET (+4.4% in AP^B). This can be explained by the fact that fewer trees are ignored, thus increasing the overall accuracy score. On the other hand, these unfiltered detections have perfectible masks, as shown by the difference in mask accuracy, which is more limited (+0.7 % in AP^M). In terms of inference speed, NMS Convex is inevitably slower than NMS, but the difference in FPS is limited (-2.5 FPS for NMS Convex, corresponding to an increased inference time of 2.5 ms per image).

B. Comparison against state-of-the-art

We train 5 different networks for comparison on SynthTree43k: Mask R-CNN [27] and Yolact [19], which have a bounding box + binary mask approach, PolarMask [20] with a polygon approach, SOLO [28] with a binary mask approach, and our ConvexMask network with a convex polygon + binary mask approach. The training parameters for the first 4 networks are exactly the same as those presented in their respective articles. For a fair comparison, no data augmentation method is used for all networks. Since SOLO and PolarMask do not produce bounding boxes in their predictions, these are calculated from the predicted masks, as was done previously for ConvexMask. The performance of the CNNs on SynthTree43k test set, with a ResNet50 or ResNet101 backbone, and their FPS, are shown in table V.

First of all, we observe that ConvexMask significantly outperforms most networks in terms of bounding box accuracy. Only Mask R-CNN achieves bounding box accuracies close to, but below, ConvexMask. The reason for this discrepancy lies in our approach, where ConvexMask focuses upstream on detecting instance extremities via convex polygon prediction. This allows us to extract a much more accurate bounding box. Secondly, in terms of mask accuracy, SOLO slightly surpasses

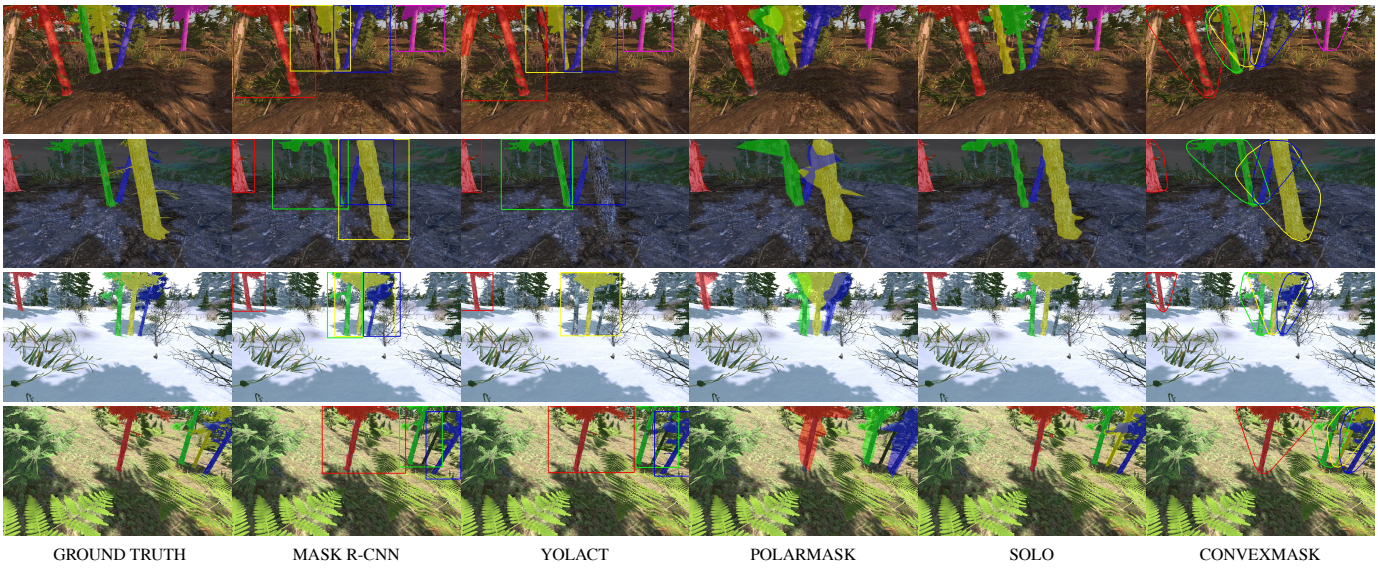


Fig. 5. Predictions of Mask R-CNN, Yolact, PolarMask, SOLO, and ConvexMask on SynthTree43k test set.

ConvexMask, both ahead of the three other CNNs. However, SOLO remains a larger network than ConvexMask, which impacts its inference speed. Moreover, ConvexMask predicts fewer tree masks correctly, as shown by the differences in performance on the AP^M_{50} metric, but these are more accurate in general, as shown by the higher scores of ConvexMask compared to SOLO on the AP^M_{75} metric. We note that the largest trees are those that suffer most from poor segmentation.

This shows that there is still room for improvement in ConvexMask's segmentation phase. Finally, ConvexMask's inference speed is on par with other state-of-the-art networks.

Fig. 5 shows CNNs prediction results on some example images from the SynthTree43k test set. First of all, ConvexMask detects tree extremities fairly accurately with the convex exterior polygon, creating a better segmentation zone compared with Yolact, reducing segmentation errors linked to trees being too close (cf. red tree in the first example image). The use of NMS Convex means that trees that are too close to each other are less ignored, as in the case of the fourth example image, compared to Yolact and PolarMask. Regarding segmentation, ConvexMask and SOLO produce similar masks. However, ConvexMask produces a more accurate segmentation than SOLO, particularly on tree trunks.

C. Prediction on real images

We test ConvexMask on real images to assess its transferability. To do this, we take as our base a ConvexMask network with a ResNet101 backbone, fully pre-trained on the COCO dataset [26], and we freeze the backbone weights in order to preserve the learning achieved on real images. We fine-tune our network solely on SynthTree43k following the $6x$ strategy [25], that is to say during 360k iterations with a learning rate decay at iterations 270k and 330k. We use several data augmentation methods, such as horizontal flipping, saturation, sheering, rotation, and image resizing, to improve model generalization. Qualitative results on synthetic images (from video sequences outside SynthTree43k) and real images are shown in Fig. 6. ConvexMask was also demonstrated on videos of synthetic and real forest scenes[1].

On synthetic data (excluding SynthTree43k), ConvexMask detects the majority of trees. However, predicted convex polygons are coarser and tend to include a larger area than the tree itself. We believe this is due to the texture of the tree leaves, which differs between the synthetic input images and those of the training dataset. Downstream, the final segmentation produces correct masks, although there is still room for improvement. Nonetheless, prediction in real-life conditions gives less qualitative results. The two real-life images in figure 6 show some interesting initial results. However, the quality of the predictions depends on many parameters: viewpoint, forest density, tree species, or brightness, to name but a few. Learning on synthetic data, therefore, shows limitations for real-world operation, as [8] and [4] relate. This transfer from synthetic to real-world is all the more complex for whole-tree detection and segmentation, requiring more varied data to adapt to the multiple forest environments possible. To remedy this, work on unsupervised learning, which does not require a fully labeled dataset, and on domain adaptation, which enables learning on synthetic data to be applied to real data, should be considered.

V. Conclusion

In this paper, we presented ConvexMask, a neural network for instance segmentation that focuses on object extremities. Through various experiments carried out on SynthTree43k, ConvexMask is able to handle several problems related to forest scenes: trees that are too close together, very complex masks, and very blurry tree extremities. The approach, based

[1]This paper has a supplementary downloadable video available at http://ieeexplore.ieee.org, provided by the authors.

Fig. 6. Prediction on synthetic (the first three on the left) and real (the last two on the right) images.

on predicting a convex polygon of the instance and then segmenting it, outperforms most state-of-the-art networks in terms of accuracy while running in real time.

However, there are still improvements needed in order to be able to predict in real-life conditions. We believe that the major improvement to be made to ConvexMask lies in its segmentation process via its protonet, which will produce more correct masks, especially for large objects. Next, work on domain adaptation is required to transfer learning to synthetic data for real-world applications. Finally, for the purposes of forest mapping and navigation, depth estimation should be integrated into ConvexMask. Through a depth modality, this new task would not only enable trees to be located in 3D space but also better differentiate trees from others from the camera's point of view.

REFERENCES

[1] X. Liu, G. V. Nardari, F. C. Ojeda, Y. Tao, A. Zhou, T. Donnelly, C. Qu, S. W. Chen, R. A. Romero, C. J. Taylor, *et al.*, "Large-scale autonomous flight with real-time semantic slam under dense forest canopy," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5512–5519, 2022.

[2] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2015.

[3] S. Nezami, E. Khoramshahi, O. Nevalainen, I. Pölönen, and E. Honkavaara, "Tree species classification of drone hyperspectral and rgb imagery with deep learning convolutional neural networks," *Remote Sensing*, vol. 12, no. 7, p. 1070, 2020.

[4] V. Grondin, F. Pomerleau, and P. Giguère, "Training deep learning algorithms on synthetic forest images for tree detection," *arXiv preprint arXiv:2210.04104*, 2022.

[5] J.-M. Fortin, O. Gamache, V. Grondin, F. Pomerleau, and P. Giguère, "Instance segmentation for autonomous log grasping in forestry operations," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 6064–6071.

[6] Z. Jiao, Y. Zhang, J. Xin, L. Mu, Y. Yi, H. Liu, and D. Liu, "A deep learning based forest fire detection approach using uav and yolov3," in *2019 1st International conference on industrial artificial intelligence (IAI)*. IEEE, 2019, pp. 1–5.

[7] D. Wang, "Unsupervised semantic and instance segmentation of forest point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 165, pp. 86–97, 2020.

[8] D. Russell, "Using drones and remote sensing to understand forests with limited groundtruth data," Master's thesis, Carnegie Mellon University, Pittsburgh, PA, August 2023.

[9] P. A. S. Mendes, A. P. Coimbra, and A. T. de Almeida, "Forest vegetation detection using deep learning object detection models," *Forests*, vol. 14, no. 9, 2023. [Online]. Available: https://www.mdpi.com/1999-4907/14/9/1787

[10] M. E. Andrada, D. Russell, T. Arevalo-Ramirez, W. Kuang, G. Kantor, and F. Yandun, "Mapping of potential fuel regions using uncrewed aerial vehicles for wildfire prevention," *Forests*, vol. 14, no. 8, 2023. [Online]. Available: https://www.mdpi.com/1999-4907/14/8/1601

[11] J. Wang, X. Chen, L. Cao, F. An, B. Chen, L. Xue, and T. Yun, "Individual rubber tree segmentation based on ground-based lidar data and faster r-cnn of deep learning," *Forests*, vol. 10, no. 9, 2019. [Online]. Available: https://www.mdpi.com/1999-4907/10/9/793

[12] A. Firoze, C. Wingren, R. A. Yeh, B. Benes, and D. Aliaga, "Tree instance segmentation with temporal contour graph," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2193–2202.

[13] K. Itakura and F. Hosoi, "Automatic tree detection from three-dimensional images reconstructed from 360° spherical camera using yolo v2," *Remote Sensing*, vol. 12, no. 6, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/6/988

[14] C. Niu, C. Newlands, K.-P. Zauner, and D. Tarapore, "An embarrassingly simple approach for visual navigation of forest environments," *Frontiers in Robotics and AI*, vol. 10, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2023.1086798

[15] J. Li, J. Liu, and Q. Huang, "Pointdmm: A deep-learning-based semantic segmentation method for point clouds in complex forest environments," *Forests*, vol. 14, no. 12, 2023. [Online]. Available: https://www.mdpi.com/1999-4907/14/12/2276

[16] D. Q. da Silva, F. N. dos Santos, A. J. Sousa, and V. Filipe, "Visible and thermal image-based trunk detection with deep learning for forestry mobile robotics," *Journal of Imaging*, vol. 7, no. 9, 2021. [Online]. Available: https://www.mdpi.com/2313-433X/7/9/176

[17] J. Lagos, U. Lempiö, and E. Rahtu, "Finnwoodlands dataset," in *Image Analysis*, R. Gade, M. Felsberg, and J.-K. Kämäräinen, Eds. Cham: Springer Nature Switzerland, 2023, pp. 95–110.

[18] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9627–9636.

[19] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9157–9166.

[20] E. Xie, P. Sun, X. Song, W. Wang, X. Liu, D. Liang, C. Shen, and P. Luo, "Polarmask: Single shot instance segmentation with polar representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 12 193–12 202.

[21] S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following," *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985.

[22] J. Sklansky, "Finding the convex hull of a simple polygon," *Pattern Recognition Letters*, vol. 1, no. 2, pp. 79–83, 1982.

[23] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[25] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[27] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[28] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li, "Solo: Segmenting objects by locations," in *European Conference on Computer Vision*, 2020, pp. 649–665.