forests

MDPI

*Article*

# Real-Time Computer Vision for Tree Stem Detection and Tracking

**Lucas A. Wells *,† and Woodam Chung**

Department of Forest Engineering, Resources and Management, College of Forestry, Oregon State University, Corvallis, OR 97331, USA
* Correspondence: lucas@silvxlabs.com
† Current affiliation: Silvx Labs, Missoula, MT 59802, USA.

**Abstract:** Object detection and tracking are tasks that humans can perform effortlessly in most environments. Humans can readily recognize individual trees in forests and maintain unique identifiers during occlusion. For computers, on the other hand, this is a complex problem that decades of research have been dedicated to solving. This paper presents a computer vision approach to object detection and tracking tasks in forested environments. We use a state-of-the-art neural network-based detection algorithm to fit bounding boxes around individual tree stems and a simple, efficient, and deterministic multiple object tracking algorithm to maintain unique identities for stems through video frames. We trained the neural network object detector on approximately 3000 ground-truth bounding boxes of ponderosa pine trees. We show that tree stem detection can achieve an average precision of 87% using a Jaccard overlap index of 0.5. We also demonstrate the robustness of the tracking algorithm in occlusion and enter–exit–re-enter scenarios. The presented algorithms can perform object detection and tracking at 49 frames per second on a consumer-grade graphics processing unit.

**Keywords:** object detection; multiple object tracking; convolutional neural network; machine vision; stereo vision



**Citation:** Wells, L.A.; Chung, W. Real-Time Computer Vision for Tree Stem Detection and Tracking. *Forests* **2023**, *14*, 267. https://doi.org/10.3390/f14020267

Academic Editor: Eduardo Tolosana

Received: 28 December 2022
Revised: 20 January 2023
Accepted: 27 January 2023
Published: 31 January 2023



## 1. Introduction

Sustainable forest management has been the central concept of managing forests throughout the past several decades and is widely recognized as an essential tool in mitigating climate change [1]. This paradigm shift has had considerable impacts on silvicultural systems and how they are implemented in many places around the world [2,3].

In order to meet a diverse set of objectives, silvicultural treatments have evolved into more elaborate, often spatially-explicit prescriptions [4]. However, these new prescriptions can undoubtedly cause an increase in implementation costs due to more time and personnel invested to layout prescriptions, the reduction in harvesting productivity, and the increased level of administration and monitoring needed to ensure environmentally compliant operations and desirable future conditions. Therefore, it has been the central focus of forest engineering and operations communities worldwide to devise forest practices that accommodate the need for complex silvicultural and harvesting prescriptions without compromising economic efficiency, safety, and environmental quality [5]. One solution is to leverage new technologies to improve forest operations' sustainability by providing improved data and augmenting equipment operators through intelligence and automation [6,7].

We have developed a vision-based automatic tree measurement and mapping system to contribute to sustainable forestry. The system consists of three components: automatic stem detection and tracking, automatic dendrometry, and vision-aided localization and mapping. The system is intended to help reduce silvicultural treatment costs by eliminating the need for individual tree marking, lessen the mental workload of equipment operators

by providing data regarding prescription compliance in real-time, and advance forest operational planning and automation through 3D visualization and mapping of trees and work environments.

This paper introduces computer vision algorithms for real-time tree detection and tracking as the first part of our vision system. Due to the complexity and details of each component, we intend to introduce the other two components in subsequent papers.

Object detection and tracking are tasks that humans can perform effortlessly in most environments. For example, humans can easily recognize trees in a forest and reassociate previously detected trees after occlusion or re-entry to the field of view. However, object detection and tracking is a complex problem for computers, and only recent research has successfully provided robust and reliable solutions. In this work, we present computer vision algorithms for two fundamental visual tasks in forestry: tree stem detection and temporal tracking in a 6-DoF camera. Object detection algorithms localize the spatial extent of some object (e.g., a tree stem) in an image. Tracking, on the other hand, maintains knowledge of the detected object through time; measurements made on a tree stem in one frame (e.g., a segmentation of the stem for diameter estimation) are available in the next frame of the video sequence. Tracking an object means that the segmentation only needs to be refined instead of segmenting the tree with no previous knowledge, which can be computationally expensive. Furthermore, a unique identifier can be maintained for a specific tree stem so that uncertainties in measurement, such as diameter, can be reduced by averaging the measurements over time.

### 1.1. Object Detection

An object detection system combines two fundamental problems in computer vision: object localization and classification. Localization seeks to recognize instances of an object in an image, typically represented by a bounding box. Classification assigns a probability that describes the likelihood of the object belonging to some class. Traditionally, object detection systems used sliding windows techniques to extract features from regions of the image and performed classification on these features using support vector machines or random forest classifiers. The most notable works on sliding window-based object detection are the Viola–Jones detector [8], the histogram of oriented gradients (HOG) detector [9], and the deformable parts-based model (DPM) [10].

In 2012, Krizhevsky et al. [11] achieved an unprecedented classification error rate on the ImageNet large-scale visual recognition challenge [12] using a deep convolutional neural network (CNN). The seminal work by [11] motivated research to integrate CNNs in object detection pipelines. CNN-based objection detection algorithms are broadly categorized into two-stage methods [13–18] and one-stage methods [19–24]. Two-stage methods perform CNN feature extraction on multiple regions in the image. One-stage methods frame detection as a regression problem and perform detection and classification by passing the entire image through the network once. We refer the reader to [25] for a review of deep learning applied to object detection.

In the context of precision forestry, more research should be dedicated to real-time tree stem detection using camera sensors due to their low costs and robustness to environmental conditions. Current research has focused on sensor fusion methods using cameras and LiDAR [26,27] or short-range structured light systems [28]. Other vision-based methods have been proposed [29–31] but rely on unrealistic assumptions about the visual contrast between the edges of the stem and the background.

### 1.2. Multiple Object Tracking

Multiple object tracking (MOT) is a well-developed problem in computer vision where the task is to detect multiple objects of some desired class or classes and track their identities through a sequence of images. Thus, MOT algorithms that do not require manual initialization of object boundaries rely on an object detection system to localize objects. Situations when the object detection system fails to detect an object or when the object

becomes occluded or out-of-sight are typically handled by predicting the object's trajectory based on previous observations. The Kalman filter [32] is a popular method for trajectory prediction in dynamical systems that are assumed to be linear-Gaussian. In situations with non-linear Gaussian motion, tracking systems appeal to the extended Kalman filter. If neither assumption can be validated, such as in cases where the camera motion is sporadic and cannot be sufficiently described by some underlying motion model, then tracking systems typically use a deterministic approach where heuristics are employed to temporally match objects across frames. The tracking algorithm presented in this work is categorized as the latter. We refer the reader to [33] for a comprehensive review of multiple object tracking.

MOT is often framed in the context of a stationary camera where multiple dynamic objects enter and exit the frame, such as in video surveillance [34,35], or as a dynamic camera tracking multiple dynamic objects, such as tracking objects from a moving vehicle [36]. Since trees are stationary objects, our problem is framed as tracking static objects from a dynamic camera, reducing the problem to estimating the motion of the camera. With knowledge of camera motion, we can make detection predictions and find an optimal assignment between predictions and new detections. However, when stems become occluded or exit and re-enter the frame, difficulties arise. The tracking algorithm presented in this chapter handles both cases.

We are not aware of any work dedicated to the problem of real-time tree stem tracking in video sequences. The work by [26] handles temporal detection matching in a simultaneous localization and mapping (SLAM) setting using the extended information filter. However, their proposed SLAM correction step is computationally expensive and unsuitable for real-time performance.

In the following sections, we provide a detailed description of the CNN-based object detector and present our methods for validation. We also describe our algorithm for tracking detected tree stems through a video sequence. Finally, we discuss our detection system's accuracy and demonstrate our tracking algorithm's capacity concerning stem identity maintenance and occlusion handling.

## 2. Materials and Methods

For data acquisition, we used a custom-built 30 cm baseline stereo camera with two high-definition image sensors (Shenzhen Ailipu Technology Co., Ltd., Shenzhen, China) in a fronto-parallel configuration. The stereo camera was calibrated using a chessboard pattern mounted on a rigid plane following methods presented in [37]. The camera was calibrated before each field visit for data collection.

In our method description below, we denote vectors as bold lowercase letters, e.g., $\mathbf{v}$, matrices as bold capital letters, e.g., $\mathbf{M}$, and scalars as lowercase italic letters, e.g., $s$. We use the notation $\|\cdot\|$ as shorthand for $\|\cdot\|_2$, i.e., the Euclidean norm. We represent images as functions, $\mathcal{I} : \Omega \to \mathbb{R}^3$ for 3-channel color images, and $\mathcal{I} : \Omega \to \mathbb{R}$ for gray-scale images where $\Omega \subset \mathbb{R}^2$ is the image domain. Sets are denoted by capital script letters, e.g., $\mathcal{A}$, and the number of elements in a set is given by $|\mathcal{A}|$.

### 2.1. Detection

Given a 3-channel color image, $\mathcal{I} : \Omega \to \mathbb{R}^3$, we seek a set of bounding boxes, $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$, such that each bounding box is parameterized as a vector, $\mathbf{b}_i = (x, y, w, h)^\mathsf{T}$ where $(x, y)$ is the center of the bounding box in the image coordinate frame, $w$ is the width of the bounding box along the $u$-axis of the image, and $h$ is the height of the bounding box along the $v$-axis of the image. A bounding box encloses a tree stem from the ground to the canopy base or the height of the image if the canopy base is not within the image domain. We use the index set $\mathcal{B}$ to index the bounding boxes, $\mathbf{b}_{i \in \mathcal{B}}$, and $|\mathcal{B}|$ to indicate the number of such boxes. A bounding box implies a sub-domain of $\mathbb{R}^2$, i.e., the interval $[x - w/2, x + w/2]$ along the $u$-axis and $[y - h/2, y + h/2]$ along the $v$-axis; thus bounding boxes are axis-aligned rectangles. For reasons that will become clear when we present the tracking algorithm, we do not restrict this domain to be a subset of the image domain, $\Omega$.

In this work, we solve for the parameters of bounding boxes representing tree stems using a modified version of the You Only Look Once (YOLO) detection algorithm [19,21,24]. At the time of this research, Redmon and Farhadi [24] was a state-of-the-art multi-class CNN-based model that performs object detection on an entire image via a single forward pass through a convolutional network. The algorithm takes a three-channel (RGB) image and resizes it according to the first layer in the network. The network's output is a tensor that encodes an object's predicted bounding box coordinates, confidence, and class probabilities (Figure 1). In the following sections, we describe the network architecture, the loss function, and the modifications we made to improve performance in the problem domain of tree stem detection.

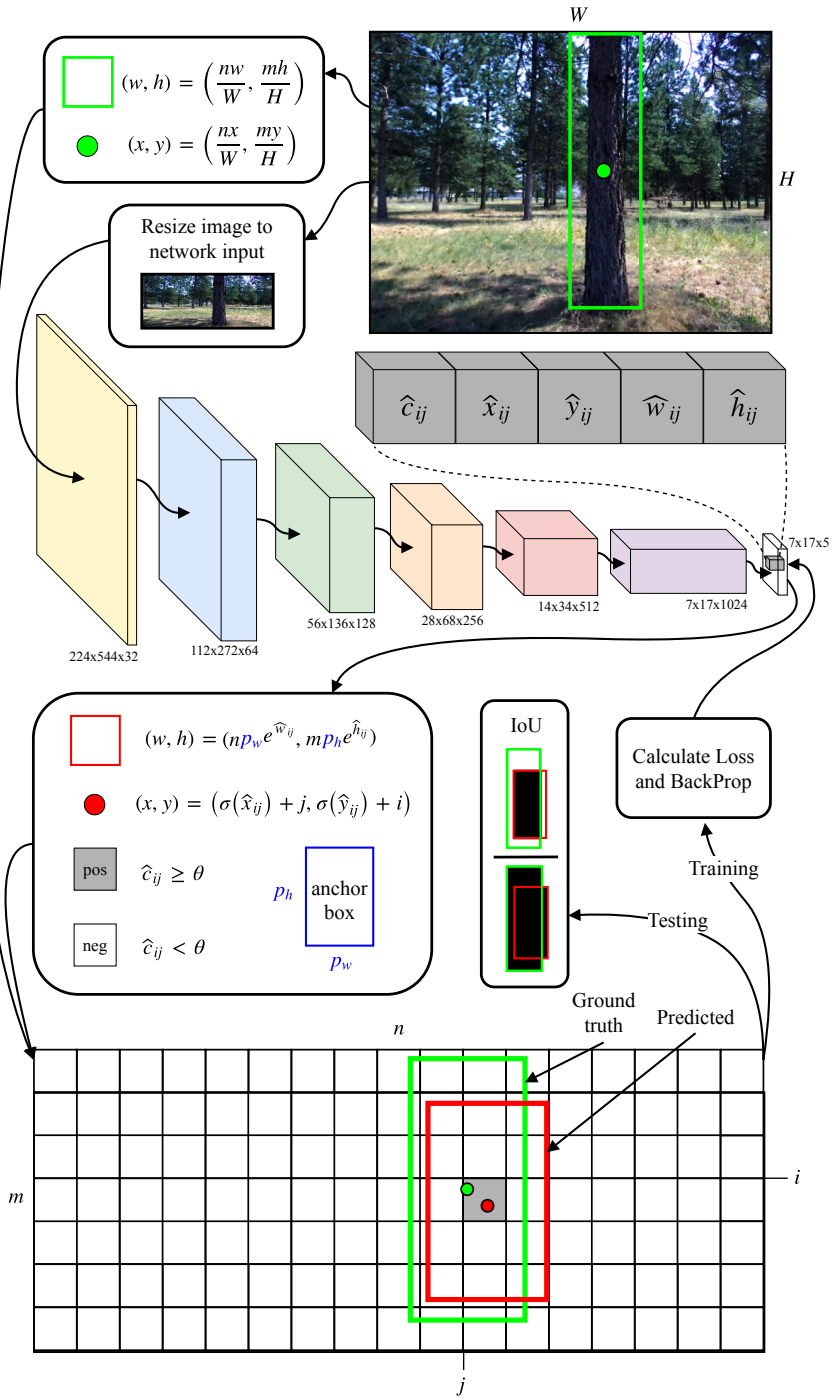



**Figure 1.** Conceptual diagram of the detection algorithm.

#### 2.1.1. Network Architecture

The original network presented in [19] used fully connected layers on top of the feature extraction layers to predict the location of the bounding box and the confidence and class probabilities. This approach led to instability in network training as the bounding box dimensions were unconstrained. This issue was improved upon in [21] by removing the fully connected layers and predicting bounding box coordinates as offsets from anchor boxes [16], resulting in a fully convolutional network that could be resized without retraining. The anchor boxes used in [21] were automatically calculated using $k$-means clustering on the ground-truth boxes in the training set. The authors heuristically selected $k = 5$ boxes for a general detection dataset including objects of drastically varying dimensions, e.g., pedestrians and cars. Therefore, each cell in the output layer predicts $k$ bounding boxes for a total of $M/2^s \times N/2^s \times k$ box predictions per image where $M$ and $N$ are the sizes of rows and columns in the input layer, respectively, and $s$ is the number of down-sampling layers in the network.

We use a single anchor box to detect tree stems and compute the dimensions by taking an average over all the ground-truth boxes in our dataset. We use a single anchor box since the aspect ratio of bounding boxes representing tree stems does not vary significantly. Futhermore, since we are only predicting one class, we do not need more than one box prediction per grid cell, given that our output resolution is sufficient. Thus, the dimensions of the output tensor for a network predicting one anchor box and one class per cell is $M/2^s \times N/2^s \times 5$. Each prediction is encoded as $(\hat{c}_{ij}, \hat{x}_{ij}, \hat{y}_{ij}, \hat{w}_{ij}, \hat{h}_{ij})$ where $\hat{c}_{ij}$ is the confidence, $(\hat{x}_{ij}, \hat{y}_{ij})$ is the center of the bounding box, $(\hat{w}_{ij}, \hat{h}_{ij})$ are the width and height, respectively, and $(i, j)$ are the indices to the output tensor along the first 2 dimensions.

We used a variant of the network presented in [21] and modified the number of filters in the final convolutional layer to accommodate one class and one anchor box. We also modified the input dimensions of the network. The original network has an input dimension of $416 \times 416$ with five down-sampling layers, resulting in an output tensor with 13 rows and 13 columns. We stretch the input layer along the $u$-axis and shrink it along the $v$-axis to increase the prediction space for tree stems distributed horizontally across the frame. Therefore, the input dimensions to our network version are 224 rows and 544 columns. We keep the number of down-sampling layers equal to 5, which produces an output tensor with dimensions $7 \times 17 \times 5$ (bottom of Figure 1), and maintain the same convolutional structure as the original network except for the pass-through layer (see [21]). We perform batch normalization on all convolutional layers and use a Leaky Rectified Linear Unit activation function for all layers except the final layer, which is linear. Table 1 describes the layers of the network in detail.

**Table 1.** Layer descriptions of the convolutional neural network for tree stem detection. Architecture modified from [21]. (Input image (rows × cols × chan): 224 × 544 × 3).

| Layer Operation | Activation Function | Filters | Size/Stride | Output Size (Rows × Cols) |
|---|---|---|---|---|
| Convolution | $\phi(x)$ † | 32 | $3 \times 3$ | $224 \times 544$ |
| Maxpool | - | - | $2 \times 2/2$ | $112 \times 272$ |
| Convolution | $\phi(x)$ | 64 | $3 \times 3$ | $112 \times 272$ |
| Maxpool | - | - | $2 \times 2/2$ | $56 \times 136$ |
| Convolution | $\phi(x)$ | 128 | $3 \times 3$ | $56 \times 136$ |
| Convolution | $\phi(x)$ | 64 | $1 \times 1$ | $56 \times 136$ |
| Convolution | $\phi(x)$ | 128 | $3 \times 3$ | $56 \times 136$ |
| Maxpool | - | - | $2 \times 2/2$ | $28 \times 68$ |
| Convolution | $\phi(x)$ | 256 | $3 \times 3$ | $28 \times 68$ |
| Convolution | $\phi(x)$ | 128 | $1 \times 1$ | $28 \times 68$ |
| Convolution | $\phi(x)$ | 256 | $3 \times 3$ | $28 \times 68$ |

**Table 1.** *Cont.*

| Layer Operation | Activation Function | Filters | Size/Stride | Output Size (Rows × Cols) |
|---|---|---|---|---|
| Maxpool | - | - | $2 \times 2/2$ | $14 \times 34$ |
| Convolution | $\phi(x)$ | 512 | $3 \times 3$ | $14 \times 34$ |
| Convolution | $\phi(x)$ | 256 | $1 \times 1$ | $14 \times 34$ |
| Convolution | $\phi(x)$ | 512 | $3 \times 3$ | $14 \times 34$ |
| Convolution | $\phi(x)$ | 256 | $1 \times 1$ | $14 \times 34$ |
| Convolution | $\phi(x)$ | 512 | $3 \times 3$ | $14 \times 34$ |
| Maxpool | - | - | $2 \times 2/2$ | $7 \times 17$ |
| Convolution | $\phi(x)$ | 1024 | $3 \times 3$ | $7 \times 17$ |
| Convolution | $\phi(x)$ | 512 | $1 \times 1$ | $7 \times 17$ |
| Convolution | $\phi(x)$ | 1024 | $3 \times 3$ | $7 \times 17$ |
| Convolution | $\phi(x)$ | 512 | $1 \times 1$ | $7 \times 17$ |
| Convolution | $\phi(x)$ | 1024 | $3 \times 3$ | $7 \times 17$ |
| Convolution | $\psi(x)$ ‡ | 5 | $1 \times 1$ | $7 \times 17$ |
| | | | | Output tensor (rows × cols × depth): $7 \times 17 \times 5$ |

† Leaky ReLU: $\phi(x) = x$ if $x > 0$ and $0.1x$ otherwise. ‡ Linear: $\psi(x) = x$.

#### 2.1.2. Training

We acquired 208 images of a ponderosa pine (*Pinus ponderosa* Douglas ex Lawson) forest in Western Montana and manually annotated 531 ground-truth bounding boxes. We used a minimum projected width threshold of $\theta = 10$ pixels to discourage the network from detecting small and distant stems. We used the following procedure to consistently annotate each tree in a frame.

1. Locate the next tree in the image.
2. Measure the projected width of the stem in the pixels and assign it to variable $p$.
3. If the projected width, $p$, is less than $\theta$, then go to step 1. Otherwise, go to step 4.
4. Position an *axis-aligned minimum bounding rectangle* enclosing the tree stem from the point of the intersection with the ground to the canopy base. The bounding box is encoded as $(x_{\min}, y_{\min}, x_{\max}, y_{\max})$.
5. Increase the maximum $x$ coordinate by $p/2$ and decrease the minimum $x$ coordinate by $p/2$, i.e., $(x_{\min} - p/2, y_{\min}, x_{\max} + p/2, y_{\max})$.
6. Convert the box extent representation to the *xy-wh* representation, $(\frac{1}{2}(x_{\min} + x_{\max}), \frac{1}{2}(y_{\min} + y_{\max}), x_{\max} - x_{\min}, y_{\max} - y_{\min})$.
7. Go to step 1.

We augmented the training set by randomly applying the following strategies: color shift by a factor of $U(0.5, 1.5)$, brightness shift by a factor of $U(0.5, 1.5)$, contrast shift by a factor of $U(0.5, 1.5)$, and reflection along the $y$-axis. We apply each augmentation with a probability of 0.5. The notation $U(a, b)$ specifies a random variate drawn from a uniform distribution on the interval $[a, b]$. We did not include image rotation in the augmentation since the camera's roll angle was always aligned with the ground plane to distribute trees horizontally in the image. Augmentation was carried out for five cycles so that our final training set had 1040 images and 2655 annotated tree stems. To train the network, we minimized the following sum of squared errors loss function adapted from [19],

$$\begin{aligned} L = \alpha \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{1}_{ij}^{\text{pos}} \left\{ \left[ \frac{nx_{ij}}{W} - \left( \sigma(\hat{x}_{ij}) + j - 1 \right) \right]^2 + \left[ \frac{my_{ij}}{H} - \left( \sigma(\hat{y}_{ij}) + i - 1 \right) \right]^2 \right\} \\ + \alpha \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{1}_{ij}^{\text{pos}} \left[ \left( \frac{nw_{ij}}{W} - np_w e^{\hat{w}_{ij}} \right)^2 + \left( \frac{mh_{ij}}{H} - mp_h e^{\hat{h}_{ij}} \right)^2 \right] \\ + \beta \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{1}_{ij}^{\text{pos}} \left[ c_{ij} - \sigma(\hat{c}_{ij}) \right]^2 + \gamma \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{1}_{ij}^{\text{neg}} \sigma(\hat{c}_{ij})^2 . \end{aligned} \tag{1}$$

We modified the original loss function presented in [19] to accommodate non-square input images, single class predictions, and single anchor box priors, and incorporated the improvements to coordinate predictions outlined in [21]. In the loss function, $\mathbf{1}_{ij}^{\text{pos}} \in \{0,1\}$ equals 1 if there is a ground-truth object in cell $(i,j)$ of the output tensor, and 0 otherwise. Conversely, $\mathbf{1}_{ij}^{\text{neg}} \in \{0,1\}$ equals 1 if there is not a ground-truth object in the $(i,j)$ cell of the output tensor, and 0 otherwise.

The first term in the loss function computes the error between the ground-truth box center, $(x,y)$, and the predicted center, $(\hat{x},\hat{y})$. The ground-truth center is scaled to the output grid dimensions, $m \times n$, according to the image width, $W$, and height, $H$. Finally, the predicted center is positioned in the output grid by first constraining the coordinates to fall in the interval $[0,1]$ with a logistic function, $\sigma(\cdot)$, then adding the position index, $(i,j)$.

The second term compares the ground-truth width and height, $(w,h)$, to the predicted width and height $(\hat{w},\hat{h})$. The ground-truth width and height are scaled to the output grid dimensions (as with the first term). The predicted width and height are calculated by raising Euler's number, $e$, to the network's prediction and multiplying with the anchor box dimensions, $(p_w,p_h)$, then scaling to the output grid dimensions. The logistic function makes the values easier for the network to learn and adds stability during training [21].

The third term calculates the error between the ground-truth confidence, $c$, and the predicted confidence, $\hat{c}$. The ground-truth confidence is simply the Jaccard index, also referred to as the intersection over union (IoU), between the ground-truth bounding box and the predicted bounding box, defined as

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}. \tag{2}$$

The index returns a value in the interval $[0,1]$, where 1 means the bounding boxes completely overlap, while 0 indicates that the boxes do not overlap at all. We use a logistic function to constrain $\hat{c}$ to fall in the interval $[0,1]$.

Finally, the last term encourages the predicted confidence to be 0 when there is no observed tree stem at the $(i,j)$ position in the output grid. The relative importance of each term in the loss function is controlled by the scalars $\alpha$, $\beta$, and $\gamma$, where $\alpha$ is the *coordinate scale*, $\beta$ is the *object scale*, and $\gamma$ is the *no object scale*. Through the empirical investigation, we determined $\alpha = 5$, $\beta = 1$, and $\gamma = 0.25$ to be adequate values for the hyper-parameters.

We trained the network for 200 epochs using a batch size of 64 images, a momentum of 0.9, and a decay rate of $5 \times 10^{-4}$. We scheduled the learning rate with an initial value of $10^{-4}$, then increased it to $10^{-3}$ at 10 epochs and $10^{-2}$ at 20 epochs. The network was trained at $10^{-2}$ for 160 epochs, then the learning rate was decreased to $10^{-3}$ for 10 epochs and decreased again to $10^{-4}$ for the final 10 epochs.

#### 2.1.3. Inference

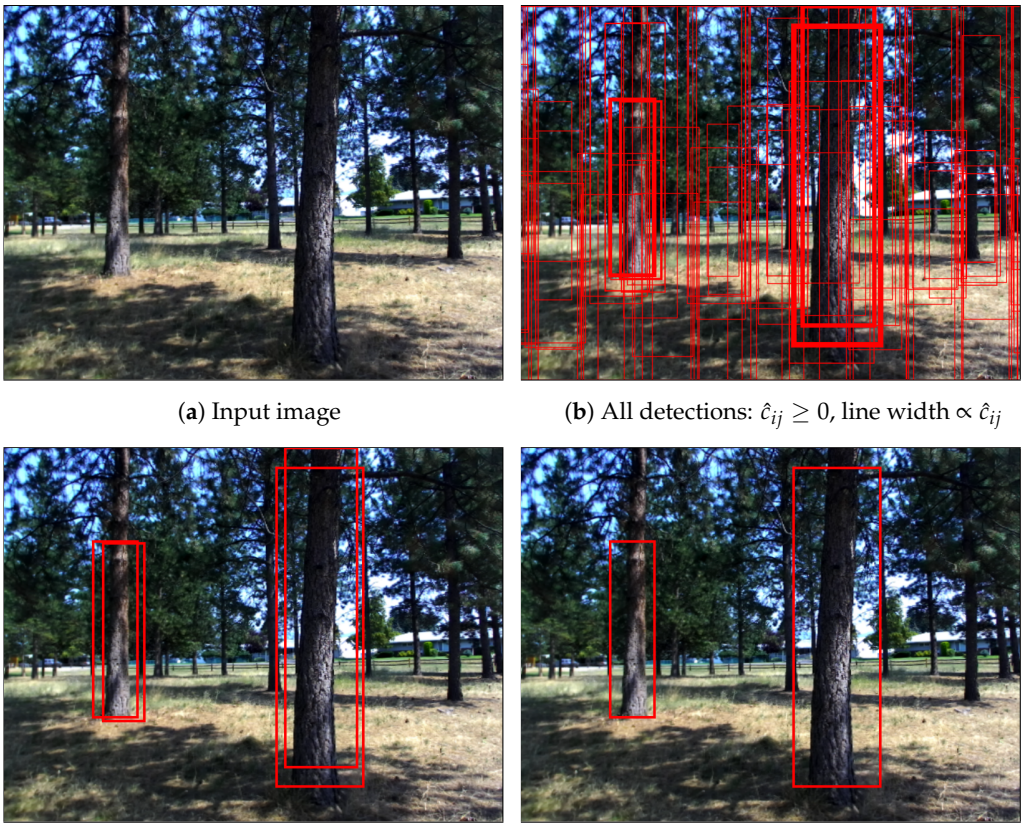
Using the trained network, we perform bounding box prediction on a new image by resizing it to increase the detection resolution along the $u$-axis, then passing it through the network and extracting the output layer, i.e., the prediction tensor. The tensor is decomposed along the depth dimension as five matrices: $\hat{\mathbf{C}}$, $\hat{\mathbf{X}}$, $\hat{\mathbf{Y}}$, $\hat{\mathbf{W}}$ and $\hat{\mathbf{H}}$ corresponding to the confidence, position along the $u$-axis, position along the $v$-axis, width along the $u$-axis, and height along the $v$-axis, respectively. Each matrix has the dimensions $m \times n$, where $m$ and $n$ are the rows and columns in the output tensor. For each row-column index to the confidence matrix, $\hat{\mathbf{C}}$, we compare the confidence score, $\hat{c}_{ij}$, against some threshold $\theta$. If the confidence is greater than or equal to the threshold, then we keep the prediction and process the coordinates with

$$\hat{\mathbf{b}}_{ij} = \begin{pmatrix} \sigma(\hat{x}_{ij}) + j - 1 \\ \sigma(\hat{y}_{ij}) + i - 1 \\ p_w e^{\hat{w}_{ij}} \\ p_h e^{\hat{h}_{ij}} \end{pmatrix} \in \mathbb{R}^4, \quad \forall\{(\{i\}_1^m \times \{j\}_1^n) : \hat{c}_{ij} \geq \theta\}\,. \tag{3}$$

Next, we perform non-maximum suppression by computing the Jaccard index between all combinations of bounding box predictions and suppressing predictions with an inferior confidence score and a Jaccard overlap with another prediction greater than $\zeta$. In this work, we use $\theta = 0.5$ for the confidence threshold and $\zeta = 0.3$ for the suppression threshold. Finally, we scale the bounding boxes to the size of the original image by

$$\mathbf{b}_{ij} = \hat{\mathbf{b}}_{ij} \odot \left( \frac{W}{n}, \frac{H}{m}, \frac{W}{n}, \frac{H}{m} \right)^{\mathsf{T}}, \tag{4}$$

where the notation $\odot$ denotes element-wise multiplication and $(W, H)$ is the width and height of the original image. We use the index set, $\mathcal{B}$, to index the scaled predictions that pass the confidence threshold and non-maximum suppression, $\mathbf{b}_{i \in \mathcal{B}} = (x, y, w, h)^{\mathsf{T}}$. Figure 2 illustrates the inference pipeline of an example image.



(**a**) Input image (**b**) All detections: $\hat{c}_{ij} \geq 0$, line width $\propto \hat{c}_{ij}$

(**c**) Threshold confidence: $\hat{c}_{ij} \geq 0.5$ (**d**) Non-maximum suppression

**Figure 2.** Inference pipeline: (**a**) Input image. (**b**) All predictions after forward pass through the network. Line widths are proportional to confidence. (**c**) Retained predictions after thresholding on the confidence. (**d**) Final prediction following non-maximum suppression.

#### 2.1.4. Testing

We tested the object detector on 103 images from the same forest where we collected the training data. We annotated the test set following the same procedure used to annotate the training set. In the test set, there were 690 ground-truth bounding boxes. We made bounding box predictions for each test image using the CNN object detector and compared

them to the ground-truth bounding boxes. We calculated precision as the number of true positives over the sum of true positives and false positives. The recall was calculated as the number of true positives over the sum of true positives and false negatives.

We used the Jaccard index to determine if a predicted bounding box corresponds to a ground-truth box. We calculated precision–recall curves at Jaccard index thresholds from 0.5 to 0.75 in 0.05 intervals. The Jaccard index threshold is denoted as $\mathcal{J}_\theta$, where $\theta/100$ is the threshold value. Thus, $\mathcal{J}_{50}$ indicates a Jaccard index threshold of 0.5. We compute each Jaccard threshold's average precision (AP) by integrating the area under the precision–recall curve.

### *2.2. Tracking*

In this section, we describe our stem tracking algorithm. We have organized this algorithm into four main components: measurement, prediction, matching, and correction. The measurement step runs the object detector described in the previous section and assigns a depth value to the bounding box. The prediction step uses the estimated ego-motion of the camera to predict the location of the detection in the current frame. The matching step finds the optimal assignment strategy between the predicted detections and new detections provided by the measurement step. Finally, the correction step updates the tracking list by replacing the predictions with matched measurements from the detector and handles the insertion and deletion of stems in the tracking list.

#### 2.2.1. Measurement

Let $M \subset \mathbb{N}$ be an index set returned by the CNN stem detector. We call this the *measurement index set*. The cardinality of this set, $|M|$, is the number of unique detections and the index $j \in M$ corresponds to a specific detection. We denote the coordinates of detection by the vector $\hat{\boldsymbol{\mu}}_{j\in M} = (x, y, w, h, d)^\mathsf{T} \in \mathbb{R}^5$, where $(x, y)$ is the center of the bounding box, $(w, h)$ is the width and height, and $d$ is the disparity assigned to the bounding box. Disparity assignment is given by the function $\delta : \mathbb{R}^2 \to \mathbb{R}^+$ defined as

$$\delta(\Psi) = \underset{i\in\{1,N\}}{\operatorname{argmax}} \sum_{\mathbf{x}\in\Psi} \mathbf{1}_i D(\mathbf{x}), \tag{5}$$

where the argument $\Psi \subset \Omega$ is a subset of the image domain bounded by $[x - \frac{w}{2}, x + \frac{w}{2}]$ and $[y - \frac{h}{2}, y + \frac{h}{2}]$ and $N$ is the number of disparity planes in the disparity function $D(\cdot)$. The indicator $\mathbf{1}_i \in \{0, 1\}$ is 1 if $i$ equals the disparity function evaluated at $\mathbf{x}$, and 0 otherwise.

To decrease the computational demand imposed by the detector, we run the detection network only when the camera is displaced by more than 30 cm or is rotated about the $x$, $y$ or $z$-axis more than 0.2 radians. We do this by first estimating the ego-motion of the camera using the direct image alignment approach based on [38]. Then, we compose the estimated ego-motion parameters until one of the above conditions is satisfied. Once the detector is executed to yield a new measurement set, the composed motion parameters are reset to zero.

#### 2.2.2. Prediction

Let $P$ be an index set to temporal detections tracked over image frames. We call this the *prediction index set*. The cardinality of this set, $|P|$, is the number of active detections, i.e., the detections currently being tracked. We encode the coordinates of a unique detection $i \in P$, as we did in $M$, with the vector $\boldsymbol{\phi}_{i\in P} = (x, y, w, h, d)^\mathsf{T} \in \mathbb{R}^5$. Each detection in $P$ at time $t$ corresponds to the detection from $M$ at time $t - n$, where $n$ is the number of frames since the prediction has been updated. We predict the center of the bounding box in frame $I_t$ by

$$\boldsymbol{\rho} = (\rho_x, \rho_y, \rho_d)^\mathsf{T} = \widetilde{\mathbf{n}}\Big(\mathbf{PT}(\boldsymbol{\xi}_i)^{-1}\mathbf{P}^{-1}\widetilde{\boldsymbol{\phi}}_i\Big), \tag{6}$$

where $\widetilde{\boldsymbol{\phi}}_i = (x, y, d, 1)^\mathsf{T}$ is the center of the bounding box in homogeneous coordinates, $\boldsymbol{\xi}_{i \in P} \in \mathfrak{se}(3)$ is the pose parameters describing the camera motion from time $t - n$ to $t$, $\mathbf{T}(\boldsymbol{\xi}_i) = \exp(\hat{\boldsymbol{\xi}}_i) \in \mathtt{SE}(3)$ is a homogeneous rigid transformation matrix, and $\widetilde{\mathbf{n}}(\cdot)$ performs homogeneous division of the coordinates, i.e., $\widetilde{\mathbf{n}}\Big((x, y, z, s)^\mathsf{T}\Big) = (x/s, y/s, z/s)^\mathsf{T}$. The camera projection matrix, $\mathbf{P}$, and its inverse, $\mathbf{P}^{-1}$, are explicitly defined as

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & c_u/f & 0 \\ 0 & 1 & c_v/f & 0 \\ 0 & 0 & 0 & b \\ 0 & 0 & 1/f & 0 \end{pmatrix}, \quad \mathbf{P}^{-1} = \begin{pmatrix} 1 & 0 & 0 & -c_u \\ 0 & 1 & 0 & -c_v \\ 0 & 0 & 0 & f \\ 0 & 0 & 1/b & 0 \end{pmatrix}, \tag{7}$$

where $(c_u, c_v)$ is the principle point in the image coordinate plane, $f$ is the focal length in pixels and $b$ is the stereo baseline in cm. Now that the box center has been projected onto the current frame, we can specify the predicted bounding box in frame $I_t$ as

$$\hat{\boldsymbol{\phi}}_i = \left(\rho_x, \rho_y, \frac{w\rho_d}{d}, \frac{h\rho_d}{d}, \rho_d\right)^\mathsf{T}, \tag{8}$$

where $w$ and $h$ are the width and height from $\boldsymbol{\phi}_i$. In summary, we warped the center of the bounding box according to the pose parameters and scaled the width and height according to the change in depth. Scaling the width and height, as opposed to warping the entire bounding box, prevents the rectangle from distortions resulting from projective projection.

For each $i \in P$ we also maintain a binary variable, $\nu_i \in \{0, 1\}$, to indicate if the predicted bounding box is *visible* in the current frame. A visible bounding box must satisfy that its domain is a subset of the image domain and it is not a subset of another bounding box in $P$. Formally, this condition is written as

$$\nu_i = \begin{cases} 1 & \text{if } \left(\hat{\boldsymbol{\phi}}_i \subset \Omega\right) \wedge \left(\hat{\boldsymbol{\phi}}_i \not\subset \hat{\boldsymbol{\phi}}_{i'}\,,\ \forall i' \in P \setminus \{i\}\right) \\ 0 & \text{otherwise} \end{cases}. \tag{9}$$

Finally, for each $i \in P$, we use a variable $\kappa_i \in \mathbb{N}$ to indicate the number of frames that a bounding box $\boldsymbol{\phi}_i$ has been visible, but has not been matched with detection from the measurement index set, $M$. We call this the *kill incrementer*. If the predicted bounding box is not matched, then we increment the variable by $\kappa_i \leftarrow \kappa_i + \nu_i$. Thus, $\kappa_i$ is only incremented when the predicted bounding box is visible.

#### 2.2.3. Matching

Given the coordinates of bounding boxes, $\hat{\boldsymbol{\mu}}_{j \in M}$, representing tree stems detected in frame $I_t$, and the predicted coordinates of previously detected stems, $\hat{\boldsymbol{\phi}}_{i \in P}$, our objective is to find the optimal temporal matching strategy. First, we construct a cost matrix, $\mathbf{C}$, where each entry in the matrix, $c_{ij}$, is calculated by

$$c_{ij} = 1 - J_{3\mathrm{D}}(\hat{\boldsymbol{\phi}}_i, \hat{\boldsymbol{\mu}}_j)\nu_i, \tag{10}$$

where $J_{3\mathrm{D}}(\cdot, \cdot)$ is the Jaccard index, as defined in Equation (2) but modified for 3-dimensional bounding boxes. A 3-dimensional bounding box is constructed by extruding the 2D bounding box by $w/2$ and $-w/2$ about the estimated depth value $z = fb/d$, again where $d$ is the assigned disparity value for the bounding box.

Next, we find the optimal matching via linear sum assignment. The cost matrix is augmented to a squared matrix by appending rows or columns, whichever is necessary, and filling with costs greater than 1. We denote entries in the augmented cost matrix with $\bar{c}_{ij}$. Using the augmented cost matrix and a binary matrix, $\bar{\mathbf{B}}$, where

$$\bar{b}_{ij} = \begin{cases} 1 & \text{if } \hat{\boldsymbol{\phi}}_i \text{ is assigned to } \hat{\boldsymbol{\mu}}_j \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

we minimize the following objective function:

$$\min \quad \sum_i \sum_j \bar{c}_{ij} \bar{b}_{ij} \tag{12a}$$

$$\text{s.t.} \quad \sum_j \bar{b}_{ij} = 1 \quad \forall i \tag{12b}$$

$$\sum_i \bar{b}_{ij} = 1 \quad \forall j \tag{12c}$$

$$\bar{b}_{ij} \in \{0, 1\} \quad \forall i, j. \tag{12d}$$

The assignment problem can be solved in $O(n^3)$, where $n = \max(\text{rows}, \text{cols})$ in $\mathbf{C}$, using the Hungarian algorithm [39]. Finally, we calculate the assignment matrix, $\mathbf{X}$, by ensuring that a candidate match in $\mathbf{B}$ has an associated cost less than some threshold,

$$x_{ij} = \begin{cases} 1 & \text{if } b_{ij} = 1 \wedge c_{ij} < \vartheta \\ 0 & \text{otherwise,} \end{cases} \tag{13}$$

where $c_{ij}$ and $b_{ij}$ are entries within the dimensions of the original cost matrix. Thus, the dimensions of $\mathbf{X}$ are equal to the dimensions of $\mathbf{C}$ prior to augmentation.

#### 2.2.4. Correction

Now that we have the optimal matching strategy, the last step is to update the prediction set and handle the unmatched detections. To update the detection coordinates in the prediction set $P$, we first stack the column vectors encoding the coordinates of all the detections in the measurement set $M$,

$$\mathbf{M} = \left( \hat{\boldsymbol{\mu}}_j, \ldots, \hat{\boldsymbol{\mu}}_{|M|} \right)^{\mathsf{T}}. \tag{14}$$

Thus, $\mathbf{M}$ is a matrix with $|M|$ rows and 5 columns, where each row encodes the coordinates of each detection $\hat{\boldsymbol{\mu}}_{j \in M}$. The matrix multiplying with $\mathbf{X}$ and transposing effectively copies the detections in $M$ to the corresponding prediction in $P$,

$$\left( \boldsymbol{\phi}_i, \ldots, \boldsymbol{\phi}_{|P|} \right)_{5 \times |P|} = (\mathbf{X}\mathbf{M})^{\mathsf{T}}. \tag{15}$$

So, the coordinates in the prediction set that were matched to the detection in $M$ are updated with new bounding box coordinates and any prediction that was not updated has the zero vector for its coordinates. An equivalent operation could be performed by searching over the assignment matrix and replacing the prediction with a measurement when a non-zero value is encountered. However, the expression in Equation (15) is more direct and concise.

If the updated coordinates are equal to the zero vector we simply retain the prediction, compose the prediction's pose parameters with the current estimate of the pose increment, $\Delta\boldsymbol{\xi}$, and update the kill incrementer. Otherwise, we reset the prediction's pose parameters and retain the current kill incrementer.

$$(\boldsymbol{\phi}_i, \boldsymbol{\xi}_i, \kappa_i) = \begin{cases} (\hat{\boldsymbol{\phi}}_i, \boldsymbol{\xi}_i \circ \Delta\boldsymbol{\xi}, \kappa_i + \nu_i) & \text{if } \boldsymbol{\phi}_i = \mathbf{0}_5 \\ (\boldsymbol{\phi}_i, \mathbf{0}_6, \kappa_i) & \text{otherwise} \end{cases}. \tag{16}$$

Next, we deactivate any prediction that has not been updated during a $\theta_{\mathbf{t}}$ displacement of the camera or has a $\kappa$ value greater than $\theta_\kappa$. In this work, we used $\theta_{\mathbf{t}} = 30$ cm and $\theta_\kappa = 0.2$ radians. The pose parameters, $\boldsymbol{\xi}_i$, represent the composed camera motion since the

last update. Thus, we can decompose the translation vector with $\left(\mathbf{R}^{(i)}|\mathbf{t}^{(i)}\right) = \exp(\hat{\boldsymbol{\xi}}_i)$ and test the following logic,

$$Q = \begin{cases} Q \cup \{i\} & \text{if } \left(\|\mathbf{t}^{(i)}\| > \theta_{\mathbf{t}}\right) \vee (\kappa_i > \theta_\kappa) \\ Q & \text{otherwise,} \end{cases} \tag{17}$$

where $Q$ is a set of indices that, according to the test, should be deactivated. We remove the indices in $P$ by set difference, $P' = P \setminus Q$, and redefine the prediction index set as $P = \{1, 2, \ldots, |P'|\}$. Finally, the indices of the prediction components are updated by

$$\boldsymbol{\phi}_i = \boldsymbol{\phi}_{f(i)}, \tag{18a}$$

$$\boldsymbol{\xi}_i = \boldsymbol{\xi}_{f(i)}, \tag{18b}$$

$$\kappa_i = \kappa_{f(i)}, \tag{18c}$$

where $f : \mathbb{N} \to \mathbb{N}$ is an ordered bijective mapping between the sets $P$ and $P'$. The last step is to check if any detections from the measurement set have not been matched. First, we perform the same operations as before for mapping predictions onto measurements through the assignment matrix, $\mathbf{X}$; however, now we do the inverse,

$$\boldsymbol{\Phi} = \left(\hat{\boldsymbol{\phi}}_i, \ldots, \hat{\boldsymbol{\phi}}_{|P|}\right)^\mathsf{T}, \tag{19}$$

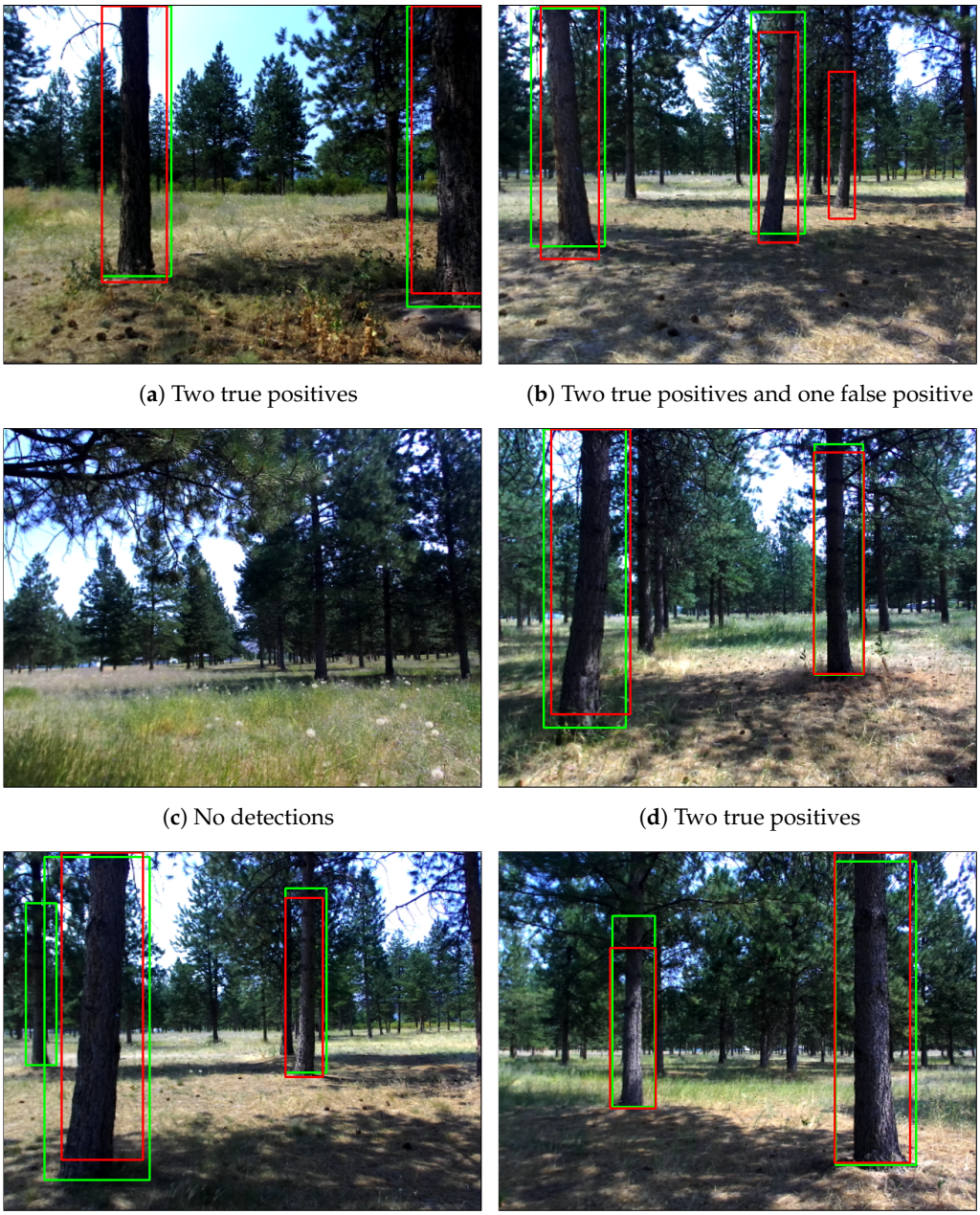
$$\left(\boldsymbol{\mu}_j, \ldots, \boldsymbol{\mu}_{|M|}\right)_{5\times|M|} = \left(\mathbf{X}^\mathsf{T}\boldsymbol{\Phi}\right)^\mathsf{T}. \tag{20}$$

If any vector $\boldsymbol{\mu}_{j\in M}$ is equal to the zero vector, then we insert the measurement into the prediction set by $P \cup \{|P| + 1\}$ and initialize the new prediction as $\boldsymbol{\phi}_{|P|+1} = \boldsymbol{\mu}_j$.

## 3. Results and Discussion

### 3.1. Detection

In Figure 3, we show images from the test set with predicted and ground-truth bounding boxes superimposed as red and green rectangles, respectively. In subfigures (a), (d), and (f), the detector performed without error since all predicted bounding boxes have a Jaccard index with the ground-truth box greater than 0.5. The detector also performed as desired in subfigure (c), as there were no false positives. In subfigure (b), there are two true positives and one false positive. In subfigure (e), there are two true positives and one false negative. We found that nearly all false negatives and positives occurred on boundary trees: trees having a projected width close to the threshold width used to determine whether a stem should be included in the training. One approach to circumvent this issue would be to annotate every tree in the image and train the network to detect all trees regardless of their projected size. However, this poses a significant burden on the network since trees far from the camera do not have the same visual features as trees close to the camera. Furthermore, the variance of bounding box dimensions would increase, demanding the inclusion of more anchor boxes during training. Therefore, we do not consider the false detections on boundary trees a significant problem since the detector will eventually correctly identify the tree stem when the projected width increases as the camera moves closer to the stem.



(**a**) Two true positives (**b**) Two true positives and one false positive

(**c**) No detections (**d**) Two true positives

(**e**) Two true positives and one false negative (**f**) Two true positives

**Figure 3.** Example detections with ground truth boxes in green and predicted boxed in red.

Figure 4 shows the precision–recall curves for Jaccard index thresholds between 0.5 and 0.75. The detector performs increasingly worse with higher Jaccard index thresholds. This is expected since perfect alignment with ground-truth bounding boxes is never realized in practice. The average precision (AP) for the detector at $\mathcal{J}_{50}$ is 89%. The mean average precision (mAP) for all Jaccard thresholds from 0.5 to 0.75 is 44%. We consider a Jaccard overlap of 0.5 sufficient for localizing tree stems. Again, we emphasize that all the true positives in Figure 3 have a Jaccard index of 0.5 or greater.

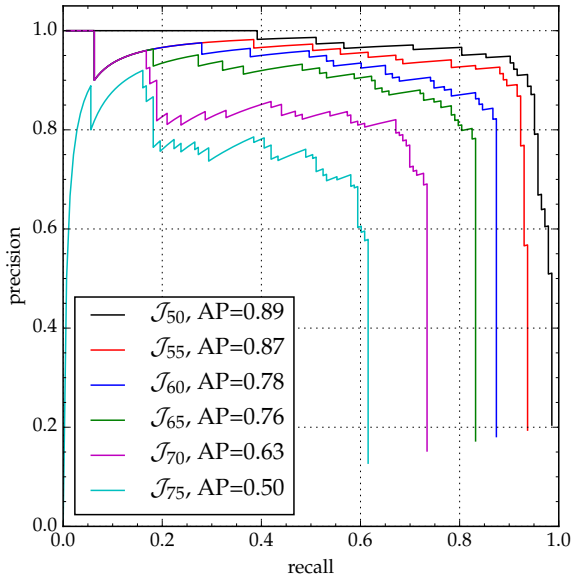



**Figure 4.** Precision–recall curves for the tree stem detection algorithm.

As mentioned earlier, the CNN is fully convolutional. Thus, we can vary the input image resolutions without retraining the network. In practice, this typically implies a speed–accuracy trade-off. In Table 2, we show the effect of input resolution on frames per second and average precision. The training resolution is shown in bold. We increased and decreased the resolution while roughly maintaining the same aspect ratio. By decreasing the resolution to 96 × 288 the detector runs at 47 fps. However, the mAP decreases to 0.21.

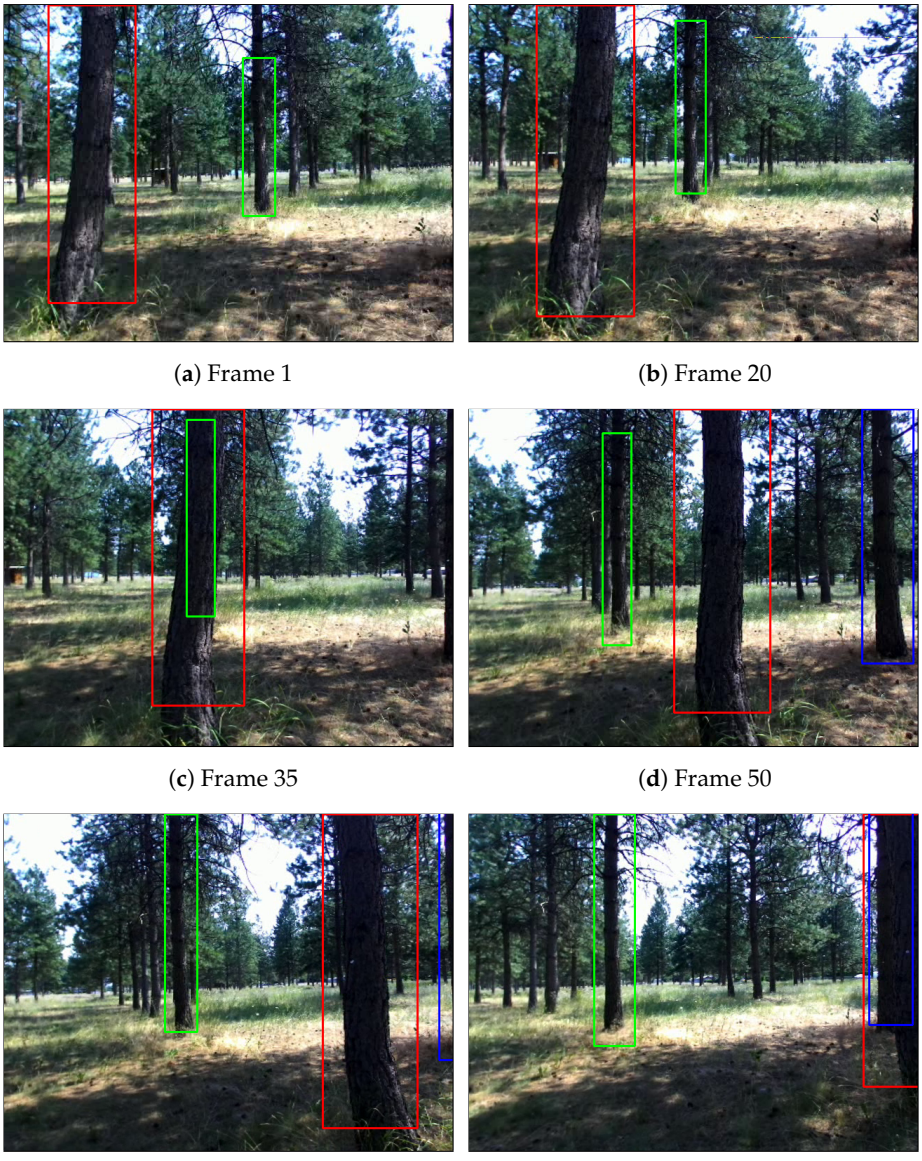
**Table 2.** Speed-accuracy trade-off for tree stem detection. The row in bold indicates the training resolution.

| **Resolution** | **FPS** | **mAP** | **Average Precision** | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | $\mathcal{J}_{50}$ | $\mathcal{J}_{55}$ | $\mathcal{J}_{60}$ | $\mathcal{J}_{65}$ | $\mathcal{J}_{70}$ | $\mathcal{J}_{75}$ |
| 96 × 288 | 47 | 0.21 | 0.59 | 0.49 | 0.35 | 0.25 | 0.17 | 0.13 |
| 128 × 352 | 38 | 0.24 | 0.63 | 0.53 | 0.43 | 0.31 | 0.23 | 0.14 |
| 160 × 416 | 26 | 0.33 | 0.79 | 0.74 | 0.66 | 0.49 | 0.31 | 0.17 |
| 192 × 480 | 23 | 0.41 | 0.85 | 0.78 | 0.68 | 0.66 | 0.53 | 0.36 |
| **224× 544** | **19** | **0.44** | **0.87** | **0.84** | **0.76** | **0.67** | **0.51** | **0.33** |
| 256 × 608 | 14 | 0.44 | 0.89 | 0.85 | 0.76 | 0.65 | 0.52 | 0.35 |
| 288 × 672 | 12 | 0.47 | 0.88 | 0.87 | 0.78 | 0.74 | 0.62 | 0.42 |

Conversely, increasing the resolution results in a marginal improvement of mAP, but significantly reduces the speed. In situations where the detector is required to execute on every frame in the video, it is advantageous to decrease the resolution and accept lower precision. On the other hand, if the camera's ego-motion is known, it is beneficial to run the detector intermittently on selected keyframes at a higher resolution and use the motion to predict the position of the bounding boxes between keyframes. Using the latter approach, we achieved a speed of 49 frames per second. This frame rate includes the processing time required to compute stereo correspondence, estimate ego-motion parameters, and run the tracking algorithm, which only takes a couple of milliseconds. We also note that the detection algorithm does not depend on a disparity map. The disparity map is only needed for the tracking algorithm. Thus, when multiple GPU devices are available, the detection algorithm and stereo correspondence can be executed simultaneously, increasing the frame rate. The run-time tests were carried out on an Nvidia GeForce GTX 780M GPU.

### *3.2. Tracking*

Figure 5 shows selected frames from a video sequence while tracking tree stems. The video was captured using a 12 cm baseline stereo camera operated at VGA resolution (480 × 640) and a frame rate of 10 Hz. The camera moved through the forest at approximately 1 $ms^{-1}$. In the first frame, the object detector localizes two tree stems. The colors represent their identities. In frame 35, the stem bounded by the green rectangle becomes occluded by the tree in the red bounding box. The location of the green stem is maintained during occlusion using the ego-motion of the camera. The stem reappears in frame 50, and a third stem is detected (shown in the blue bounding box). In the next frame, frame 85, the tree bounded by the blue rectangle is no longer in the field of view; however, its position is still correctly predicted. Finally, in frame 110, the tree bounded by the blue rectangle is partially occluded by the tree bounded by the red rectangle. Although the detection algorithm cannot identify these stems as two separate instances, their correct identities and locations in the image are maintained since they were detected as individual stems in previous frames.



(**a**) Frame 1 (**b**) Frame 20

(**c**) Frame 35 (**d**) Frame 50

(**e**) Frame 85 (**f**) Frame 110

**Figure 5.** Selected frames from a video sequence in which tree stems are tracked. The colors of the bounding boxes indicate the stem's identity.

Except for the measurement step, which relies on the stem detection algorithm, and ego-motion estimation, all other computations involved in the tracking algorithm are efficient, taking only 1–2 milliseconds to execute. As discussed in the previous section, the detection algorithm runs 19 frames per second. However, since the tracking algorithm only runs the detector when the camera is displaced or rotated by a certain amount, we can achieve a higher frame rate. In order to estimate the ego-motion of the camera, it was necessary to perform stereo matching in each frame. We used a real-time GPU implementation of semi-global stereo matching presented in [40] and achieved a frame rate of 90 FPS. Furthermore, we performed ego-motion estimation on 120 $\times$ 160 resolution images and obtained a frame rate of over 200 FPS. Therefore, stereo matching and ego-motion estimation can be executed at approximately 60 FPS. We found the average run time for the tracking algorithm, including stem detection, stereo matching, and ego-motion estimation, to be approximately 49 FPS on the GTX 780M GPU.

We do not present quantitative results for the tracking algorithm. We did not encounter any instances of switching identities or new identities assigned to stems in the active tracking list in our dataset. Although we expect this to be an issue in dense forests, we consider intermittent identity confusion to be non-detrimental to the system since this algorithm is intended to reduce variance through the temporal accumulation of measurements (such as diameter) and to provide real-time visual information during operation. However, an incorrect temporal association must be avoided in mapping systems, as it can cause the system to fail completely. We will address the data association issue in a future publication that presents an approach for large-scale forest mapping.

Although it is common practice in the MOT field to compare results to existing tracking algorithms, we were not successful in finding an implementation that works in this specific problem domain where the camera is moving with a known ego-motion and the tracked objects are stationary and intermittently detected by a separate process. We are actively working on gathering a publicly-available dataset that covers many forest types and structures to aid in developing and comparing object detection and tracking algorithms in forest operations.

## 4. Conclusions

This paper presents visual processing algorithms for tree stem detection and tracking in video sequences. We consider these algorithms fundamental in automatic tree measurement and mapping systems. Object detection–tracking is a well-developed subdiscipline in computer vision; however, few studies have applied vision-based detection and tracking to forestry. To our knowledge, the work presented in this paper is the first attempt at real-time detection and tracking in forested environments. Tree stem detection and tracking, coupled with our earlier work on ground plane extraction and breast height estimation [41], provides a foundation for a system to perform real-time and automatic dendrometry.

A notable limitation of our approach is species classification; tree species play an essential role in tree selection during prescription development and implementation. Since our dataset was collected in a forest stand composed of a single species, we did not consider species classification in our detection algorithm. The CNN detection network presented here can be extended to multi-class detection with minimal modifications. However, it is difficult to know the efficacy of species classification using only the tree's stem as different species can exhibit similar visual features and patterns on their bark. We consider species classification a critical step in moving this work forward.

## 5. Patents

US Patent No. US011481972B2: Method of performing dendrometry and forest mapping.

**Author Contributions:** Conceptualization, L.A.W. and W.C.; methodology, L.A.W. and W.C.; software, L.A.W.; validation, L.A.W. and W.C.; formal analysis, L.A.W.; investigation, L.A.W. and W.C.; resources, L.A.W.; data curation, L.A.W.; writing—original draft preparation, L.A.W.; writing—review and editing, L.A.W. and W.C.; visualization, L.A.W.; supervision, W.C.; project administration,

W.C.; funding acquisition, W.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the U.S. Forest Service National Technology and Development Program under contract number 16CS-1113-8100-017.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Rametsteiner, E.; Simula, M. Forest certification—An instrument to promote sustainable forest management? *J. Environ. Manag.* **2008**, *67*, 87–98. [CrossRef]
2. Nagel, L.M.; Palik, B.J.; Battaglia, M.A.; D'Amato, A.W.; Guldin, J.M.; Swanston, C.W.; Janowiak, M.K.; Powers, M.P.; Joyce, L.A.; Millar, C.I.; et al. Adaptive Silviculture for Climate Change: A National Experiment in Manager-Scientist Partnerships to Apply an Adaptation Framework. *J. For.* **2017**, *115*, 167–178. [CrossRef]
3. Seidl, R.; Rammer, W.; Lexer, M.J. Adaptation options to reduce climate change vulnerability of sustainable forest management in the Austrian Alps. *Can. J. For. Res.* **2011**, *41*, 694–706. [CrossRef]
4. Achim, A.; Moreau, G.; Coops, N.C.; Axelson, J.N.; Barrette, J.; Bédard, S.; Byrne, K.E.; Caspersen, J.; Dick, A.R.; D'Orangeville, L.; et al. The changing culture of silviculture. *For. Int. J. For. Res.* **2021**, *95*, 143–152. [CrossRef]
5. Marchi, E.; Chung, W.; Visser, R.; Abbas, D.; Nordfjell, T.; Mederski, P.S.; McEwan, A.; Brink, M.; Laschi, A. Sustainable Forest Operations (SFO): A new paradigm in a changing world and climate. *Sci. Total Environ.* **2018**, *634*, 1385–1397. [CrossRef]
6. Chung, W.; Lyons, K.; Wells, L.A. *Innovations in Forest Harvesting Technology*; Burleigh Dodds Science Publishing: Cambridge, UK, 2019; pp. 489–512.
7. Lindroos, O.; Mendoza-Trejo, O.; La Hera, P.X.; Ortíz Morales, D. *Advances in Using Robots in Forestry Operations*; Burleigh Dodds Science Publishing: Cambridge, UK, 2019; pp. 233–260. [CrossRef]
8. Viola, P.; Jones, M. Robust real-time object detection. *Int. J. Comput. Vis.* **2001**, *57*, 87.
9. Dalal, N.; Triggs, B. Histograms of Oriented Gradients for Human Detection. In Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; pp. 886–893. [CrossRef]
10. Felzenszwalb, P.F.; Girshick, R.B.; McAllester, D.; Ramanan, D. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 1627–1645. [CrossRef]
11. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems 25 (NIPS 2012), Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
12. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]
13. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'14), Columbus, OH, USA, 23–28 June 2014; pp. 580–587. [CrossRef]
14. He, K.; Zhang, X.; Ren, S.; He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [CrossRef]
15. Girshick, R. Fast R-CNN. In Proceedings of the 2015 International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
16. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Proceedings of the Advances in Neural Information Processing Systems 28 (NIPS 2015), Montreal, QC, Canada, 7–12 December 2015; pp. 91–99.
17. Dai, J.; Li, Y.; He, K.; Sun, J. R-FCN: Object Detection via Region-based Fully Convolutional Networks. In Proceedings of the Advances in Neural Information Processing Systems 29 (NIPS 2016), Barcelona, Spain, 5–10 December 2016; pp. 379–387.
18. Lin, T.Y.; Dollár, P.; Girshick, R.B.; He, K.; Hariharan, B.; Belongie, S.J. Feature pyramid networks for object detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17), Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
19. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, Real-time object detection. *arXiv* **2015**, arXiv:1506.02640.
20. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In Proceedings of the 2016 European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 11–14 October 2016.
21. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. *arXiv* **2016**, arXiv:1612.08242.
22. Lin, T.; Goyal, P.; Girshick, R.B.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. *arXiv* **2017**, arXiv:1708.02002.
23. Fu, C.Y.; Liu, W.; Ranga, A.; Tyagi, A.; Berg, A.C. DSSD: Deconvolutional single shot detector. *arXiv* **2017**, arXiv:1701.06659.
24. Redmon, J.; Farhadi, A. YOLOv3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.

25. Zhao, Z.; Zheng, P.; Xu, S.; Wu, X. Object detection with deep learning: A review. *arXiv* **2018**, arXiv:1807.05511.
26. Auat Cheein, F.; Steiner, G.; Perez Paina, G.; Carelli, R. Optimized EIF-SLAM algorithm for precision agriculture mapping based on stems detection. *Comput. Electron. Agric.* **2011**, *78*, 195–207. [CrossRef]
27. Shalal, N.; Low, T.; McCarthy, C.; Hancock, N. Orchard mapping and mobile robot localisation using on-board camera and laser scanner data fusion–Part A: Tree detection. *Comput. Electron. Agric.* **2015**, *119*, 254–266. [CrossRef]
28. Juman, M.A.; Wong, Y.W.; Rajkumar, R.K.; Goh, L.J. A novel tree trunk detection method for oil-palm plantation navigation. *Comput. Electron. Agric.* **2016**, *128*, 172–180. [CrossRef]
29. Lu, Y.; Rasmussen, C. Tree trunk detection using contrast templates. In Proceedings of the 2011 18th IEEE International Conference on Image Processing, Brussels, Belgium, 11–14 September 2011; pp. 1253–1256. [CrossRef]
30. Shao, L.; Chen, X.; Milne, B.; Guo, P. A novel tree trunk recognition approach for forestry harvesting robot. In Proceedings of the 2014 9th IEEE Conference on Industrial Electronics and Applications, Hangzhou, China, 9–11 June 2014; pp. 862–866. [CrossRef]
31. Shao, L.; Mu, Y.; Liu, J.; Dong, G.; Liu, H.; Guo, P. The trunk of the image recognition based on BP neural network. In Proceedings of the 2014 IEEE International Conference on Mechatronics and Automation, Tianjin, China, 3–6 August 2014; pp. 1800–1805. [CrossRef]
32. Kalman, R.E. A New Approach to Linear Filtering and Prediction Problems. *J. Basic Eng.* **1960**, *82*, 35–45. [CrossRef]
33. Luo, W.; Xing, J.; Milan, A.; Zhang, X.; Liu, W.; Zhao, X.; Kim, T.K. Multiple Object Tracking: A Literature Review. *arXiv* **2014**, arXiv:1409.7618.
34. Wang, X. Intelligent multi-camera video surveillance: A review. *Pattern Recognit. Lett.* **2013**, *34*, 3–19. [CrossRef]
35. Koller, D.; Weber, J.; Malik, J. *Robust Multiple Car Tracking with Occlusion Reasoning*; Technical Report UCB/CSD-93-780; EECS Department, University of California: Berkeley, CA, USA, 1993.
36. Betke, M.; Haritaoglu, E.; Davis, L.S. Real-time multiple vehicle detection and tracking from a moving vehicle. *Mach. Vis. Appl.* **2000**, *12*, 69–83. [CrossRef]
37. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 1330–1334. [CrossRef]
38. Lucas, B.D.; Kanade, T. An Iterative Image Registration Technique with an Application to Stereo Vision. In Proceedings of the 7th International Joint Conference on Artificial Intelligence, IJCAI'81, Vancouver, BC, Canada, 24–28 August 1981; Morgan Kaufmann Publishers Inc.: Burlington, MA, USA, 1981; Volume 2, pp. 674–679.
39. Kuhn, H.W.; Yaw, B. The Hungarian method for the assignment problem. *Naval Res. Logist. Quart* **1955**, *2*, 83–97. . [CrossRef]
40. Hernandez-Juarez, D.; Chacón, A.; Espinosa, A.; Vázquez, D.; Moure, J.C.; López, A.M. Embedded Real-time Stereo Estimation via Semi-Global Matching on the GPU. In Proceedings of the International Conference on Computational Science 2016, ICCS 2016, San Diego, CA, USA, 6–8 June 2016; pp. 143–153. [CrossRef]
41. Wells, L.A.; Chung, W. Evaluation of Ground Plane Detection for Estimating Breast Height in Stereo Images. *For. Sci.* **2020**, *66*, 612–622. [CrossRef]