Published in Image Processing On Line on 2022–12–11.
Submitted on 2021–03–08, accepted on 2022–09–09.
ISSN 2105–1232 © 2022 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2022.338

Ant Colony Optimization for Estimating Pith Position on Images of Tree Log Ends

Rémi Decelle[1], Phuc Ngo[1], Isabelle Debled-Rennesson[1], Frédéric Mothe[2], Fleur Longuetaud[2]

[1] Université de Lorraine, CNRS, LORIA, F-54000 Nancy, France (remi.decelle@loria.fr)
[2] Université de Lorraine, AgroParisTech, INRAE, SILVA, F-54000 Nancy, France

Communicated by Lara Raad *Demo edited by* Lara Raad

Abstract

The pith location is one of the most important features to detect in order to determine the quality of wood. Indeed, it allows to extract other important features. In this paper, we address the problem of pith detection on images of wood cross-sections. Taking such images can be done at little cost and with a high resolution. However, contrary to computed tomographic images, digital images exhibit disturbances like sawing marks, dirt or ambient light variations which make difficult the image analysis. Few studies have focused on such images. Furthermore these studies do some prior segmentation or cropping before the detection. We propose an approach for estimating the pith location without any requirements. Our method is based on an ant colony optimization algorithm. It is a probabilistic approach for solving this task. We validate our algorithm on images of Douglas fir captured after harvesting. The efficiency of this algorithm has been demonstrated by performance comparisons with other approaches. Experiments show an accurate and fast estimation and the algorithm could be used in real time, at sawmill environment or in forest, with a smartphone.

Source Code

A C++ implementation of Ant Colony Optimization for Pith estimation is available at the web page of this article[1]. Compilation and usage instructions are included in the `README.txt` file of the archive. The associated online demo is accessible through the web site.

Keywords: agent-based method; Fourier transform; Hough transform; wood quality

[1] https://doi.org/10.5201/ipol.2022.338

1 Introduction

In this paper, we focus on an automatic tool to extract an important feature for wood quality estimation. The quality of wood influences both its price and use. The wood quality is defined by some properties among which:

1. mechanical resistance;
2. dimensional stability. Wood is hygroscopic, meaning that it can gain or lose moisture from the surrounding air, which could be source of trouble;
3. durability, which is the ability to resist to fungi and insects without chemical treatments;
4. aesthetic, including color, texture and aspect features.

Some of these properties are related to other features that are easily observable on wood. For instance annual ring width, that indicates how fast the tree has grown, has a direct effect on wood density that itself is related to mechanical strength, shrinkage, durability and sometimes color. For wood quality estimation, one of the key features to detect is the pith location. More precisely, pith location allows to analyse tree rings easily. Moreover, assessing how far the pith is located from the geometric center of a cross-section (i.e. eccentricity) is an important information on wood quality. It is an indicator of the presence of reaction wood[2] that may affect mechanical properties [1].

The problem of detecting automatically the pith has already been studied in the literature [8, 13, 6, 16, 2, 17, 12]. Many methods have been developed for X-ray computed tomographic (CT) images [8, 13, 6, 2, 7]. A CT scanner is an investment too heavy for most sawmills. Extracting external information on log quality by using low cost RGB cameras would be valuable for scientists and timber industry. Contrary to CT images, digital images exhibit disturbances like dirt or sawing marks which make the analysis harder. Some images of our dataset are shown in Figure 1. The dataset is split into 2 subsets: **Besle** imageset and **BBF** imageset. Both subsets include the same log ends shot with different lighting conditions (see Section 5.1 for more details).

Figure 1: Some examples of **Besle** dataset (the first two rows) and **BBF** dataset (the last row).

[2]Reaction wood is a special wood that helps the tree to straighten vertically.

In this context, Norell et al. [16], Schraml et al. [17] and Kurdthongmee et al. [12] have proposed algorithms for pith detection on digital images of untreated wood cross-sections. Recently, Deep Neural Networks (DNNs) have been trained in order to address this problem [11]. On the other hand, state-of-the-art approaches based on RGB images rely on tree rings analysis. They can be described with three steps:

1. Estimate normals from tree ring local orientations. Indeed, close to the pith, tree rings describe approximately concentric circles (with the pith at the center).

2. Add normals in an accumulation space (similar to Hough transform [5]).

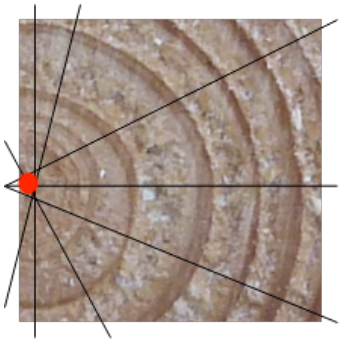
3. Extract the pith from the accumulation space.

Figure 2 exemplifies an accumulation space. Six normals are plotted and added in the accumulation space. The accumulation space is equivalent to a voting grid.

This method has shown its robustness over the others (e.g. morphological or grayscale analysis) according to Wei et al. [19].

In order to compute local orientation, one can use Fourier transform [17], linear symmetry, quadrature filters [16] or histograms of oriented gradient [12]. Then, different strategies can be used to add normals into the accumulation space, such as selecting all normals or selecting wisely those to accumulate. Finally, a strategy has to be defined to extract the pith from the accumulation space. For instance, one can assume that the pith is located at the pixel with maximum accumulation [17, 12] or at the barycenter of a set of points above a given accumulation threshold [16].

Among the three steps, the normal accumulation is the hardest task as it is difficult to determine which normals to accumulate. In this paper, we propose to use *Ant Colony Optimization* (ACO) as strategy to accumulate normals. ACO is an algorithm inspired by the behavior of ant species. Ants deposit pheromones that help other ants of the colony to make the best choice in their goal. *Ant system* was the first ACO Algorithm [4]. Since then, a large number of ACO algorithms have been developed to address various problems like edge detection in images [18, 14]. In our ACO-based algorithm, the main idea is to consider each normal as a set of pheromones to deposit. Our implementation also has to work without any prior segmentation. To the best of our knowledge, no such method has been developed so far. Indeed, the algorithms proposed in [16, 17, 12] are robust and give the pith location with a millimetric precision, but they need some prior assumptions. Norell et al. [16] manually segmented the wood section. Both Kurdthongmee et al. [12] and Schraml et al. [17] assumed that the wood section was the center of their images and segmented the wood section. The robustness, accuracy and speed of our method will be assessed and compared to the literature. The results show the good performance of our method.



1		1					1	1
1		1				1	1	
1	1			1	1			
1	1	1	1					
3	1							
6	2	1	1	1	1	1	1	1
2		1	1					
1	1			1	1			
1		1				1	1	
1			1				1	1

Figure 2: Left: six normals plotted according to tree ring local orientation. The pith is marked by the red circle. Right: the accumulation space according to these normals. Each accumulation cell is 10×10 pixels.

Section 2 explains the preprocessing of images and the proposed ACO method. Section 3 describes the algorithms. Section 4 details the implementation, including requirements, settings and some examples of use. Section 5 shows experimentation and results. The conclusion is made in Section 6.

2 Proposed Method

In this section, we present the main steps of our method for pith estimation. Figure 3 shows the three main steps of our proposed method: the preprocessing, which aims at both removing sawings marks and enhancing tree rings; the local orientation of tree ring; and, finally, our ACO implementation for pith estimation.

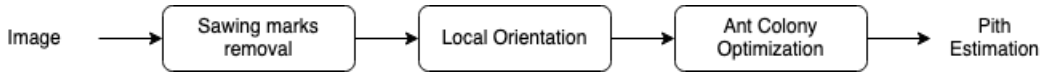


Figure 3: The three main steps of the proposed method.

2.1 Preprocessing

We first apply a preprocessing. It aims at reducing both time computation and noise. The preprocessing consists in converting RGB to grayscale, downsampling and removing sawing marks.

2.1.1 Grayscale and Downsampling

We convert the RGB image to a grayscale image. Conversion to grayscale is done by using the formula $0.299R + 0.587B + 0.114G$, where R, G and B are respectively red, green and blue color channels. Then, downsampling is done using a factor δ and bilinear interpolation. It keeps the image ratio. This step aims at reducing both noise between annual tree rings and calculation time. The noise is caused by both unsharpened sawing tools and our acquisition device which takes high resolution images.

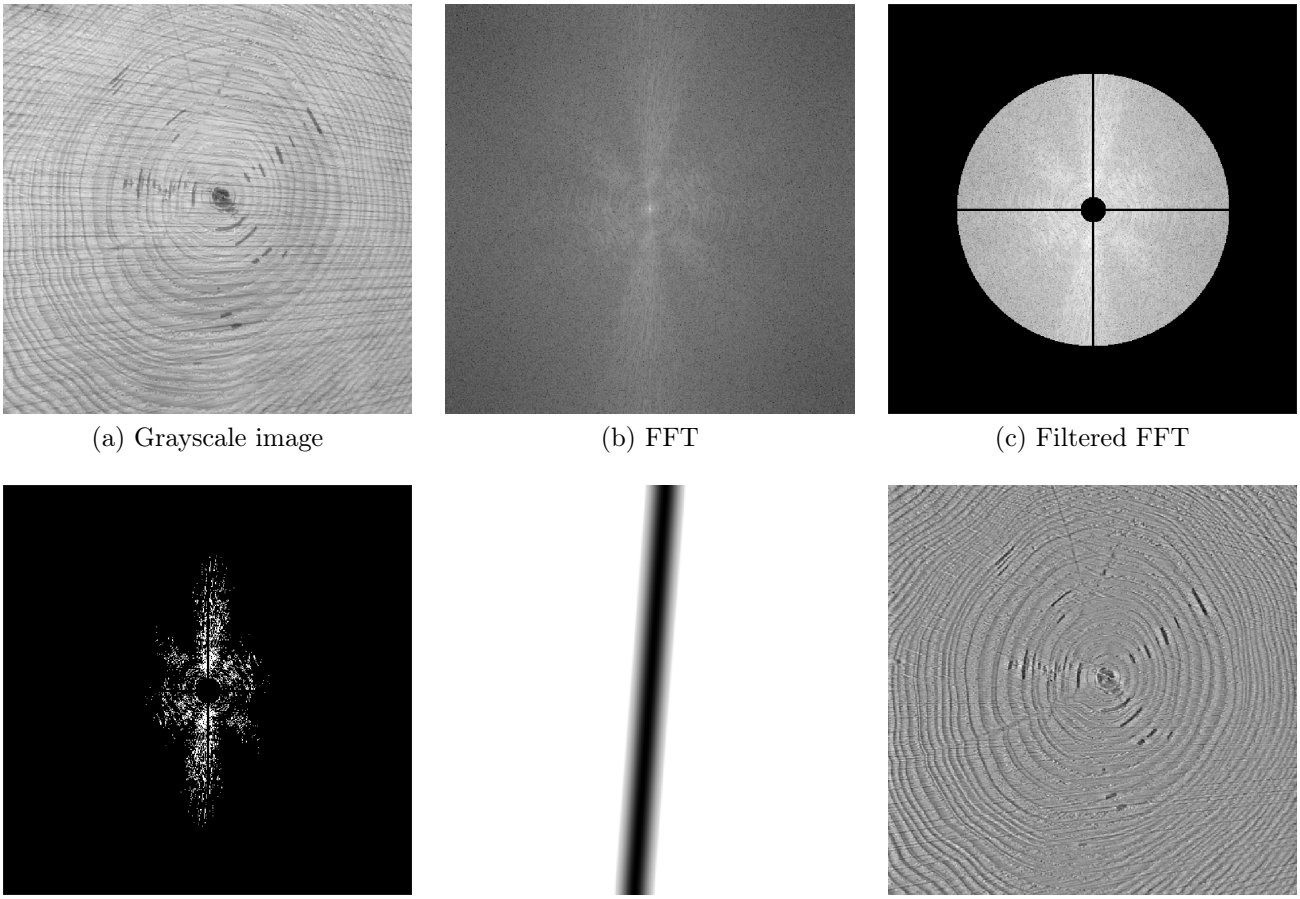
2.1.2 Sawing Marks Removal

One of the main disturbances for pith detection on untreated sections is the presence of sawing marks on the tree log image (see Figure 4a). Since pith detection relies on tree ring orientation, those marks may increase errors by interfering with tree rings. We propose a simple way to remove them. This is similar to the one proposed by Norell [15].

One can observe that sawing marks are generally aligned in a circular sector and not always evenly spaced (see Figure 4a). This repetitive pattern suggests that filtering in Fourier domain is a suitable choice to reduce them. Indeed, sawing marks correspond to a line in the Fourier spectrum passing through the center with a direction perpendicular to those marks. Along this line, the energy in Fourier spectrum will be high. By reducing the energy along that line and converting filtered spectrum back to spatial domain, sawing marks can be removed or at least reduced (see Figure 4f).

We first compute a Fast Fourier Transform (FFT). In order to reduce noise due to background, we filter the Fourier spectrum with a band-pass filter and remove both horizontal and vertical lines which often are regions with a high energy due to shifting (see Figure 4b and 4c).

We then threshold the filtered Fourier spectrum keeping only values above λ times the maximum value (see Figure 4d). The DC component (the (0,0) Fourier coefficient) is excluded from this maximum. This gives a point cloud on which we apply principal component analysis (PCA) to



(a) Grayscale image (b) FFT (c) Filtered FFT

(d) FFT after thresholding (e) Detected line (f) Preprocessed image

Figure 4: Preprocessing steps: (a) Grayscale image, (b) FFT, (c) Filtered FFT, (d) FFT after thresholding, (e) Detected line, (f) Resulting image. The lighter the pixel, the larger its value.

estimate two orthogonal directions. With λ, we can control the sawing marks removal. Pronounced sawing marks give high value in Fourier domain, then λ will have a high value. Once the main eigenvector is retrieved, we create a filter defined by a line along the estimated direction and passing through the Fourier spectrum center. The line is convolved with a Gaussian filter of standard deviation σ (see Figure 4e). Sawing marks are not strictly parallel lines, that's why we introduce σ as a margin of error in sawing marks. It thickens the line to remove in Fourier domain.

Finally, the line filter is pixelwise multiplied by the original Fourier spectrum. The result is transformed back into a spatial image using the inverse FFT (see Figure 4f).

Sometimes the main directions in the point cloud are not orthogonal. This leads PCA to misestimate the wanted directions since the method looks for orthogonal directions. Figures 5a and 5b illustrate this problem. Red arrows indicate the desired directions, and blue arrows indicate those estimated by PCA. This is the reason why we only apply sawing removal if

$$\frac{\nu_1 - \nu_2}{\nu_1} > 0.66$$

where ν_1 is the eigenvalue of the principal component and ν_2 the eigenvalue of the second component. If the value is below that threshold it may have another direction. As we cannot know which one to remove, we will not apply sawing removal. It may also indicate that sawing marks are not very pronounced, then they will not disturb the pith estimation. The value of 0.66 has been chosen empirically.

2.2 Local Orientation

Actually, tree rings can be described by concentric circles with the pith at the center. The key point of the algorithm is to accumulate normals to tree ring tangents in a robust way. That is the reason why, after removing sawing marks, we need to determine local orientation at each pixel. For this, Norell and Borgefors [16] used quadrature filters and linear symmetry for estimating local orientation. Schraml and Uhl [17] estimated it by FFT analysis, and Kurdthongmee et al. [12] retrieved it by using histograms of oriented gradients (HoGs) [3].

In this paper, we use a smooth gradient based method. The orientation is computed in a local window with Gaussian filters as an approximation of the gradient. Then, another smoothing is applied. It prevents from abruptly changing orientations. It was used by Hong et al. [9] for assessing the local orientation of pixels in fingerprints images which are close to tree ring images (light and dark area alternating).

The first step of this method is to compute the gradient $(\partial_x(I), \partial_y(I))$ of the image I. We choose to compute the gradient with a Sobel operator associated with a Gaussian filter F of standard deviation σ_g and a size of $6\sigma_g + 1$. Then the orientation I_θ is given by $I_\theta = \arctan(\partial_x(I), \partial_y(I))$. However, the solution has some drawbacks. Firstly, it is non-linear and presents discontinuity. Secondly, in spite of the Gaussian filtering this method is sensitive to noise. And, since in our case the orientation is circular, the average orientation between two angles like 175° and 5° should be not 90° as an arithmetic average would suggest but 0°.

Kass and Witkin [10] proposed a way to solve this last problem. The idea is to double the angles and encode the orientation in a vector

$$\mathbf{d} = [r\sin(2I_\theta), r\cos(2I_\theta)]$$

where r is proportional to the strength of the orientation estimate (i.e. the squared norm of the gradient: $\partial_x(I)^2 + \partial_y(I)^2$). An average is done in a local window to obtain a more robust estimate of $\mathbf{d}$. For that, a second Gaussian filter G of standard deviation σ_b and a size of $6\sigma_b+1$ is applied. Using trigonometric formulae and $I_\theta = \arctan(\partial_x(I), \partial_y(I))$, the orientation vector $\mathbf{d}$ can be simplified. From that, Hong et al. [9] derived the above idea for computing image orientation. They computed several gradient estimates to retrieve the dominant orientation

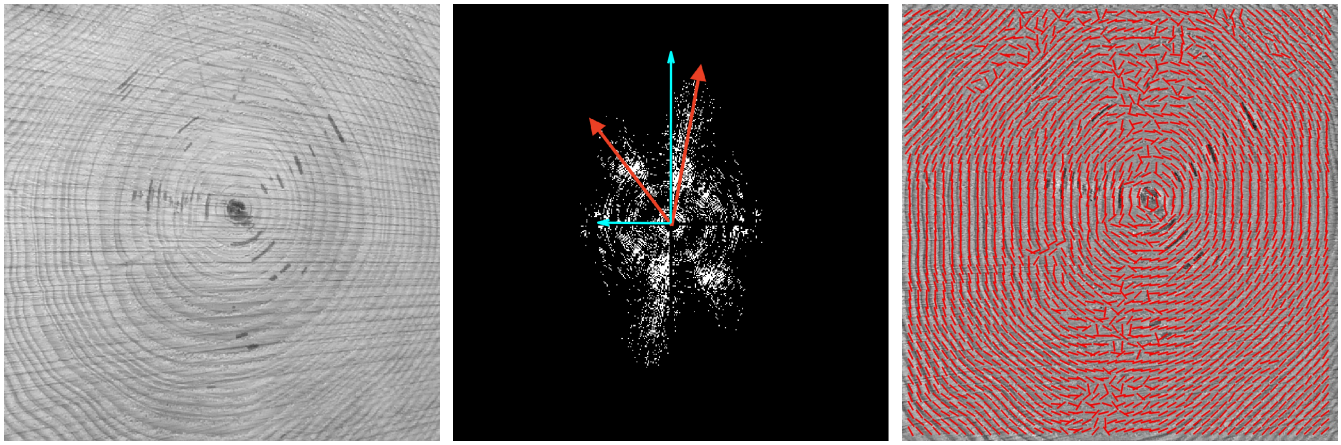
$$\begin{aligned} I_\theta &= \frac{1}{2}\arctan(2Z, X - Y) + \frac{\pi}{2}, \\ Z &= \partial_x(I)\partial_y(I) * G, \\ X &= \partial_x(I)^2 * G, \\ Y &= \partial_y(I)^2 * G. \end{aligned}$$

These three quantities are normalized before applying the arctan. Furthermore, since we want the tangent to the orientation, the additional value $\frac{\pi}{2}$ is omitted.

Figure 5c shows the preprocessed image and local orientation estimated by this method. Because of sawing marks removal, a few tree ring local orientations are disturbed. Even so, it will not prevent ACO from determining the pith position.

2.3 Ant Colony Optimization

Once local tree ring orientations have been estimated, we need to accumulate normals. Near the pith, tree rings can be described by concentric circles with the pith at the center. Thus, normals should accumulate in a way that the pith is at the position with the most accumulations. Instead of designing a heuristic to accumulate normals in a robust way (i.e. rejecting normals with wrong



(a) Grayscale image (b) Point cloud (c) Orientation estimation

Figure 5: Input image (left) and its corresponding point cloud (middle) where the two main directions (red arrows) in Fourier Domain are non-orthogonal. PCA cannot estimate these directions. Result of local orientation of tree rings (red lines) (right). Some orientations are disturbed due to sawing marks removal.

orientation), we rely on ant colony optimization algorithms (ACO) [4, 18]. In the proposed method, an ant has a position in the image from which normals will be accumulated. Each ant performs two actions: moving according to a pheromone map τ and depositing pheromone, which increases thevalue in this pheromone map τ. This map will be our accumulation space. Then, ants will move more often to areas near the pith (i.e. areas with a high quantity of pheromones) and accumulate normals near the pith. And ants will move less often to areas away from the pith (i.e. areas with low quantity of pheromones) and accumulate less often normals away from the pith. Let us now describe how ACO works.

Let K^2 be the number of ants and $\tau^{(t)}$ the pheromone map at time t. The two main questions in the ACO algorithm are how to establish the probabilistic transition matrix, denoted by $\mathbf{p}^{(t)}$, and how to update the pheromone map $\tau^{(t)}$. Our choices for each question are presented in detail below.

2.3.1 Initialization Process

K^2 ants are placed in a uniform grid on the preprocessed image. Each ant can move freely on the pheromone map. At the beginning, the initial value of each element of the pheromone map $\tau^{(0)}$ is set to a random number between 0 and 1 by a normal distribution. A normal distribution seems to accelerate the convergence of the algorithm. An element in matrix $\tau^{(t)}$ stands for $m \times m$ pixels in the preprocessed image. In other words, if the preprocessed image $\mathbf{I}$ is of size $H \times W$ then $\tau^{(t)}$ has $\frac{H}{m}$ by $\frac{W}{m}$ elements. In the following, matrix $\tau^{(t)}$ is considered to be in (u, v) coordinate and the image is in (x, y) coordinate. The relation between a pixel (x, y) and an element (u, v) is: $(u, v) = (\lfloor \frac{x}{m} \rfloor, \lfloor \frac{y}{m} \rfloor)$. A high value for m reduces both computational time and accuracy. A low value increases computational time but it increases accuracy.

2.3.2 Constructing the Probabilistic Transition Matrix

An ant moves randomly with a certain probability. The probability $\mathbf{p}_{u,v}$ for an ant to move to the element (u, v) is defined by

$$\mathbf{p}_{u,v} = \frac{(\tau_{u,v})^{\alpha}(\eta_{u,v})^{\beta}}{\sum_{i,j}(\tau_{i,j})^{\alpha}(\eta_{i,j})^{\beta}}, \tag{1}$$

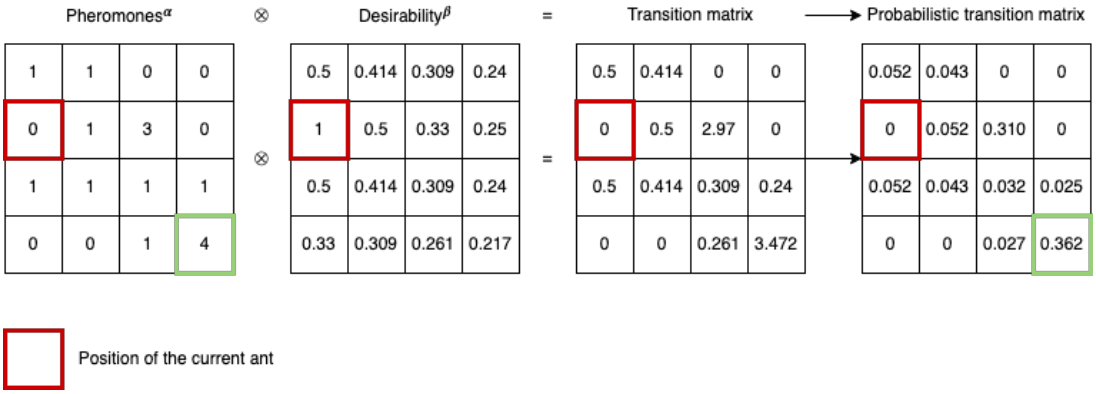


Figure 6: Example of probabilistic transition matrix computation. The red square is the current ant's position. After computation, the ant has a probability of 36.2% to move to the element with the most pheromones (green squares). The result is obtained with $\alpha = 2$ and $\beta = 1$.

where $\tau_{u,v}$ is the amount of pheromone at element (u, v), $\eta_{u,v}$ is the desirability of element (u, v), α and β are respectively parameters to control the influence of $\tau_{u,v}$ and $\eta_{u,v}$. The desirability at an element (u, v) is equal to the inverse distance between this element and the element (a, b) where the ant is. So, the desirability is equal to

$$\eta_{u,v}(a, b) = \frac{1}{\sqrt{(u-a)^2 + (v-b)^2} + 1}.$$

The desirability can be seen as a weighting of the pheromone map. It aims to ensure ants having a higher probability to move toward local maxima and not toward a global one which could be a wrong pith estimation. The ratio $\frac{\alpha}{\beta}$ makes it possible to modify the behavior of ants. A high value leads ants to move more quickly to the pheromone peaks. A low value leads ants to continue to explore areas in the image.

2.3.3 Pheromone Map Update

The pheromone map is updated as follows

$$\tau^{(t+1)} = (1 - \gamma)\tau^{(t)} + \sum_{k \in \{0,\dots,K^2-1\}} \rho_k^{(t)}, \tag{2}$$

where $\rho_k^{(t)}$ is the matrix of deposited pheromones by the k^{th} ant for the current iteration t and γ is the rate of pheromone evaporation. The evaporation rate eliminates wrong accumulations over time. It aims at removing local peaks in pheromones. The higher γ, the faster pheromones are removed. The update only occurs once all ants have moved.

Figure 6 shows how to compute $\mathbf{p}$ for an ant according to τ (pheromone map) and η (desirability matrix).

2.3.4 Pheromone Deposit

Let us now describe how an ant deposits pheromones. For each ant k we estimate $\rho_k^{(t)}$ before moving it. But, it is only added to the pheromones matrix $\tau^{(t)}$ after moving all the ants. We compute $\rho_k^{(t)}$ as follows.

Each ant is the center of an image block cluster. The cluster consists of $n \times n$ blocks and each block has a size of $\omega \times \omega$ pixels. Parameter ω acts as a smoothing because the local orientation could

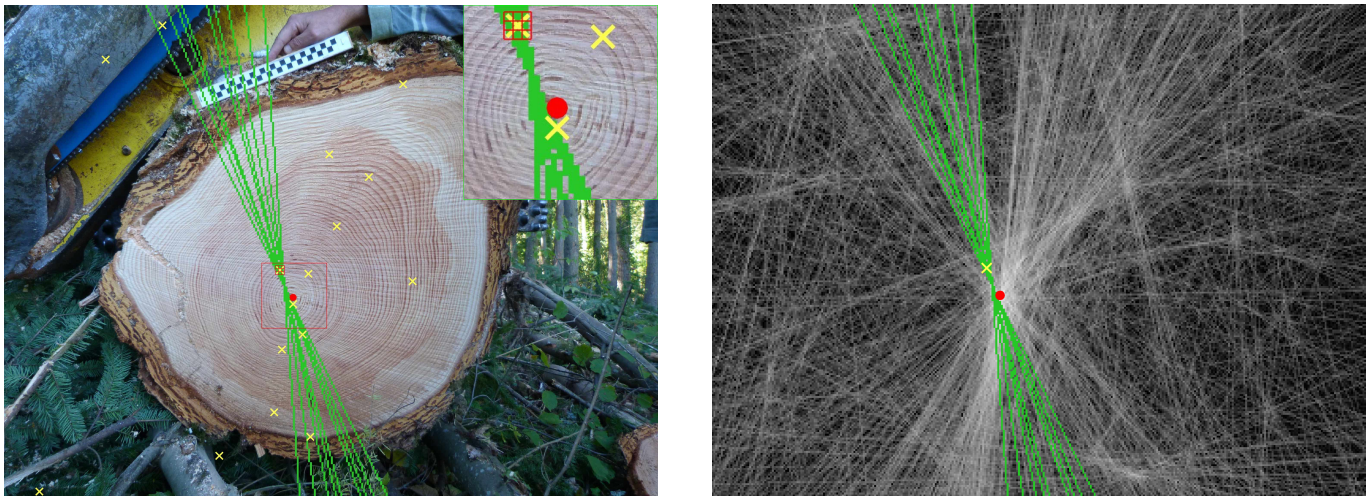


Figure 7: On the left, image coordinates with the ants in yellow crosses, the pith estimation in red, a cluster of 3 blocks of size 8 as the red grid ($n = 3$ and $\omega = 8$) and normals (according to the cluster) in green lines. The top right is a zoom on the red grid and current pith estimation. On the right, same objects in pheromones coordinates. One pixel in pheromones coordinates is equal to 5 pixels in image coordinates ($m = 5$).

have been miscalculated. A small value is tantamount to consider that the orientation has been correctly calculated. Indeed, for each block of a cluster, the median value of the local orientation (see Section 2.2) is taken as the block orientation. These steps are all performed in preprocessed image coordinates. The position (x, y) of the block center is converted into the closest element (u, v). As a reminder, the relation between pixels and elements is: $(u, v) = (\lfloor \frac{x}{m} \rfloor, \lfloor \frac{y}{m} \rfloor)$, where m is the factor between τ and the preprocessed image. Then a line is drawn according to the orientation and passing through (u, v). All elements of $\rho_k^{(t)}$ on the line are incremented by 1.

Figure 7 shows, on the left, an ant (yellow cross) with its cluster of blocks (red grid). For each block of the cluster, a line is drawn in green according to its orientation. The current pith estimation is in the red circle. On the right, same objects (with same color) are represented in (u, v) coordinates. The pith estimation is the red circle. The background is the pheromones matrix τ.

2.3.5 Extraction of the Pith Position

To extract from the matrix τ the pith position, we take the barycenter of all the pixels above $\kappa * \max(\tau)$. Taking the maximum is less robust than the barycenter of the highest values. The higher κ, the more sensitive to small variations the pith estimation.

We introduce an early stopping criterion. At each iteration, we compute the distance between the current and the last pith estimation. The algorithm stops as soon as the average of the last five distances falls below a threshold ε. The algorithm stops after a maximum of N iterations.

3 Algorithm

Hereafter, we detail the different algorithms for the steps described previously. A complexity analysis of the proposed method is given at the end of the section.

3.1 Preprocessing and Local Orientation Algorithms

The first algorithm (Algorithm 1) is the preprocessing for sawing marks removal (see Section 2.1). The second algorithm (Algorithm 2) presents the computation of local orientations (see Section 2.2).

Algorithm 1: Sawing marks removal algorithm (see Section 2.1.2)

Input: Input image I
The threshold for Fourier spectrum λ
Standard deviation for a Gaussian smoothing σ
Output: Preprocessed image J
Variables: g is a Gaussian kernel of size $2\lceil 2\sigma \rceil + 1$ and standard deviation σ
(x_c, y_c) the position of the center

$F \leftarrow \text{FFT}(I)$ // *include shift the zero-frequency component to the center*
$G \leftarrow F$
Remove both centered horizontal and vertical lines in G
Apply a low-pass filter to G
$\mu \leftarrow \max(G)$
$G \leftarrow G \geq \lambda \times \mu$ // *Pixel-wise comparison*
Let P be the positions of the threshold pixels
$\nu_1, \nu_2 \leftarrow \text{eigenvalues}(\text{PCA}(P))$ // *Eigenvalues of PCA(P)*
$(V_x, V_y) \leftarrow \text{eigenvector}(\text{PCA}(P))$ // *The main eigenvector of PCA(P)*
$\zeta \leftarrow \frac{\nu_1 - \nu_2}{\nu_1}$ // *The certainty*
if $\zeta > 0.66$ **then**
  $D \leftarrow \text{Accumulate}(\{(x_c, y_c, \text{atan2}(\frac{V_y}{V_x}))\})$ // *See Algorithm 5*
  $D \leftarrow D * g$ // $*$ *denotes the convolution product*
  $F \leftarrow F \times D$ // *Pixel-wise multiplication*
$J \leftarrow \text{IFFT}(F)$ // *The inverse FFT*

3.2 Ant Colony Optimization Algorithm

The main idea of our ACO algorithm is provided in Algorithm 3. The reader may refer to [4] for a detailed description of the ACO algorithm. Algorithm 4 details the ants' moves and the computation of the pheromones deposit map (see Section 2.3.4).

Then, Algorithm 5 describes how to accumulate lines in a map (or an image) given a set of points and orientations in order to obtain a map accumulation. The accumulation is done as follows. For each couple of point and orientation, we first compute the coefficients (a,b) of the line. Then, we compute the intersection between the line and the map, which is a rectangle $h \times w$. Thus, we intersect the line with lines $y = 0$, $y = h - 1$, $x = 0$ and $x = w - 1$. We keep the two points that are at the edge of the map. Finally, we calculate the Bresenham line passing through these two points and we increment by one the value of the map along the Bresenham's line.

Finally, Algorithm 6 presents the pith extraction from the pheromone map (see Section 2.3.5).

Algorithm 2: Local Orientation Algorithm (see Section 2.2)

Input: Input image I
Standard deviation for gradient smoothing σ_g
Standard deviation for co-variance smoothing σ_b
Standard deviation for output smoothing σ_s

Output: Local orientation image I_θ

Variables: F is a Gaussian kernel of size $6\sigma_g + 1$ and standard deviation σ_g
G is a Gaussian kernel of size $6\sigma_b + 1$ and standard deviation σ_b
H is a Gaussian kernel of size $6\sigma_s + 1$ and standard deviation σ_s
S_x Sobel kernel for x-derivative
S_y Sobel kernel for y-derivative

1 $f_x \leftarrow F * S_x$ and $f_y \leftarrow F * S_y$ // $*$ *denotes the convolution product*
2 $\partial_x \leftarrow I * f_x$ and $\partial_y \leftarrow I * f_y$
3 $X \leftarrow \partial_x^2 * G$, $Y \leftarrow \partial_y^2 * G$, $Z \leftarrow 2(\partial_x \partial_y * G)$ // *Pixel-wise multiplication*
4 $\Theta_x \leftarrow \frac{Z}{\sqrt{Z^2+(X-Y)^2}}$
5 $\Theta_y \leftarrow \frac{X-Y}{\sqrt{Z^2+(X-Y)^2}}$ // *Pixel-wise operations*
6 $\Theta_x \leftarrow \Theta_x * H$
7 $\Theta_y \leftarrow \Theta_y * H$
8 $I_\theta \leftarrow \frac{1}{2}\arctan(\Theta_x, \Theta_y)$ // *Pixel-wise operations*

3.2.1 Complexity Analysis

The ACO algorithm complexity is dominated by two factors: the number of ants and the size of the image. As a reminder, the number of ants is K^2 and the size of the pheromones is $\left\lfloor \frac{H}{m} \right\rfloor, \left\lfloor \frac{W}{m} \right\rfloor$. As we focus on complexity, we omit the factor m. First, each ant can deposit up to $\sqrt{H^2 + W^2}$ pheromones. Then, the pheromones map, which is of size $H \times W$, has to be updated for each ant. Finally, each ant moves to its new position in $\mathcal{O}(1)$. This yields a time complexity of $\mathcal{O}(K^2 \left[\sqrt{H^2 + W^2} + HW + 1\right])$.

An easy way to decrease the ACO computation time is to parallelize the computation of one ant update. Indeed, ants move and update the pheromones map independently of each other.

Algorithm 3: Ant colony optimization for normal accumulation

Input: The number of ants $K \times K$
The maximum number of iterations N
The number of block clusters around an ant $n \times n$
The size of each block cluster $\omega \times \omega$ pixels

Output: Pith estimation $\xi^{(t)}$ at iteration t

Variables: τ^t: The pheromone map at iteration t
ρ_k^t: The deposited pheromone map of the k^{th} ant at iteration t

Pre-process image I // *See Section 2.1 and Algorithm 1*
Compute the local orientations I_θ // *See Section 2.2 and Algorithm 2*
$\delta_y \leftarrow \frac{\text{height}(I)}{K}$
$\delta_x \leftarrow \frac{\text{width}(I)}{K}$
for $i = 0 \dots K-1$ **do**
  for $j = 0 \dots K-1$ **do**
    $a \leftarrow j\delta_x + \lfloor \frac{\delta_x}{2} \rfloor$
    $b \leftarrow i\delta_y + \lfloor \frac{\delta_y}{2} \rfloor$
    Add an ant at position (a, b)
Initialize the pheromone map τ^1
for $t = 1 \dots N$ **do**
  for $k = 1 \dots K^2$ **do**
    Let (a, b) be the position of the k^{th} ant
    $\rho_k^{(t)} \leftarrow \text{Move}(a,b,I_\theta, \tau^t, n, \omega)$ // *See Section 2.3.2 and Algorithm 4*
  $\tau^{(t+1)} \leftarrow (1-\gamma)\tau^{(t)} + \sum_{k=1}^{K} \rho_k^{(t)}$ // *Update the pheromone map, see Section 2.3.3*
  $\xi^{(t+1)} \leftarrow \text{estimate\_pith}(\tau^{(t+1)})$ // *Compute the new pith position, see Section 2.3.5 and Algorithm 6*
  $d_{t+1} \leftarrow \left\| \xi^{(t+1)} - \xi^{(t)} \right\|_2$ // *Distance between the current and the last pith position*
  if $t \geq 5$ **then**
    $a_t \leftarrow \frac{1}{5} \sum_{k=t-3}^{t+1} d_k$ // *Compute the average of the last five distances*
    if $(a_t < \varepsilon)$ **then**
      Break // *Early-stopping criteria*
Return $\xi^{(t)}$

Algorithm 4: Move and compute deposited pheromone map

Input: (a, b) the position of the k^{th} ant
Estimated local orientations I_θ
The pheromone map $\tau^{(t)}$ at iteration t
The number of block clusters around an ant $n \times n$
The size of each block cluster $\omega \times \omega$ pixels

Output: The deposited pheromone map $\rho_k^{(t)}$ of the k^{th} ant at iteration t

Variables: Height H of the pheromone map $\tau^{(t)}$
Width W of the pheromone map $\tau^{(t)}$
B the list of the $n \times n$ block clusters of size $\omega \times \omega$ centered at (u, v) in I_θ

$L \leftarrow \{\}$
foreach $h \in B$ **do**
  $\theta \leftarrow \text{median}(h)$ // *Median value of h*
  $(u, v) \leftarrow$ Center of h
  $L \leftarrow L \cup \{(u, v, \theta)\}$
$\rho_k^{(t)} \leftarrow \text{Accumulate}(L, H, W)$ // *See Algorithm 5*
foreach $(u, v) \in \tau_k^t$ **do**
  $\eta_{u,v} \leftarrow \frac{1}{\sqrt{(u-a)^2+(v-b)^2}+1}$ // *Compute the desirability matrix η of the ant, see Section 2.3.2*
foreach $(u, v) \in \tau_k^t$ **do**
  $\mathbf{p}_{u,v} \leftarrow \frac{(\tau_{u,v}^t)^\alpha (\eta_{u,v})^\beta}{\sum_{i,j} (\tau_{i,j}^t)^\alpha (\eta_{i,j})^\beta}$ // *Compute the probabilistic transition matrix $\mathbf{p}$ of the ant*
$p_u \leftarrow \sum_j \mathbf{p}_{u,j}$
$x \leftarrow \text{random}(p_u)$ // *Select a random index with probability p_u*
$p_v \leftarrow \mathbf{p}_{x,v}$ // *Probabilities of the column x of $\mathbf{p}_{u,v}$*
$y \leftarrow \text{random}(p_v)$ // *Select a random index with probability p_v*
Move the ant to the position (x, y)
Return $\rho_k^{(t)}$

Algorithm 5: Draw an accumulation of a set of lines

Input: A set of lines L
The dimension of the map (h, w)
Output: A the accumulation map

$A \leftarrow 0$ *// Initialize to zero-matrix*
foreach $(i, j, o) \in L$ **do**
  $a \leftarrow tan(o)$ *// Compute the line L : y = ax+b*
  $b \leftarrow j - ai$
  $V \leftarrow (-1, -1)$, $W \leftarrow (-1, -1)$, $G \leftarrow (-1, -1)$ and $H \leftarrow (-1, -1)$
  if $|a| > 1e^{-12}$ **then**
    $V \leftarrow (-\frac{b}{a}, 0)$ *// Intersection between L and y=0*
    $W \leftarrow (\frac{h-1-b}{a}, h-1)$ *// Intersection between L and y=h-1*
  if $|a| < 1e^{12}$ **then**
    $H \leftarrow (0, b)$ *// Intersection between L and x=0*
    $G \leftarrow (w-1, a(w-1)+b)$ *// Intersection between L and x=w-1*
  Let (P_0, P_1) be the points that are at the edge of the map (among V, W, H, G)
  $S \leftarrow \overline{P_0P_1}$ *// Segment P_0P_1 using Bresenham algorithm*
  for $x \in S$ **do**
    $A(x) \leftarrow A(x) + 1$
Return A

Algorithm 6: Estimate pith position

Input: Pheromone map τ^t at iteration t
Threshold for the barycenter κ $(0 \leq \kappa \leq 1)$
Output: Pith estimation $\xi^{(t)}$ at iteration t

$m \leftarrow \max(\tau^t)$
$X \leftarrow \tau^t \geq \kappa \times m$ *// Pixel-wise comparison*
$M_{00} \leftarrow \sum_{i,j} X_{ij}$, $M_{10} \leftarrow \sum_{i,j}(iX_{ij})$ and $M_{01} \leftarrow \sum_{i,j}(jX_{ij})$ *// M_{ij} are moments of X*
$\xi^{(t)} \leftarrow (\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}})$
Return $\xi^{(t)}$

4 Implementation

The proposed method has been implemented in C++. The implementation is available at

http://www.ipol.im/pub/art/2022/338/

An online IPOL demo is also available for testing without any installation.

https://ipolcore.ipol.im/demo/clientApp/demo.html?id=338

Hereafter, we explain how to compile and use the C++ implementation.

4.1 Requirements

Our C++ implementation only relies on the open source library *OpenCV*[3] (OPEN Computer Vision). We provide in addition a CMakeList to compile the code. It requires at least **cmake 3.4**[4].

4.2 Input and Output

The input of the method is an image. The following file formats are supported: jpg, bmp, png, tiff and pgm. There are two outputs. The first one is the input image on which the pith position is marked with a red cross. The second one is a CSV file with the pith position, time computation and the seed used for the random generator (required for ants movement).

4.3 Setting the Parameters

The proposed method for pith estimation requires the following parameters:

- δ the reduction factor of the image (see Section 2.1.1);
- K the number of ants in a row;
- α the control of the pheromone influence in Equation (1);
- β the control of the heuristic influence in Equation (1);
- γ the evaporation rate in Equation (2);
- m how many pixels an element in the matrix τ stands for;
- n the size of the blocks cluster (see Section 2.3.4);
- ω the size in pixels of a block (see Section 2.3.4);
- κ threshold for the barycenter (see Section 2.3.5);
- ε threshold to early stop the algorithm;
- N the maximum number of iterations.

[3] https://opencv.org/
[4] https://cmake.org/download/

4.4 Compilation and Use

To compile the code with **cmake**:

- Create a build directory: **mkdir build**
- Move to the directory: **cd build**
- Precompile the project: **cmake ..**
- Then compile the project: **make**

This procedure is also detailed in the provided *README.md*. The compilation creates an executable called **AntColonyPith** which can be used as follows:

```
./AntColonyPith --input=path_to_image [list_of_parameters]
```

```
./AntColonyPith --input=path_to_image --parameters=path_to_parameters.json
```

Some images are provided in the directory **samples**. Hereafter, several usage examples:
To run the program on **harvest.jpeg** with default parameters.

```
./AntColonyPith --input ../src/samples/harvest.jpeg
```

To run the program on **harvest.jpeg** with 10×10 ants, $\alpha = 1.0$ and without animation.

```
./AntColonyPith --input ../src/samples/harvest.jpeg --ant=10 --alpha=1.0 --animated=false
```

More details about the options are given in the command line helper.

```
./AntColonyPith --help
```

5 Experiments and Results

5.1 Dataset

We evaluated our method with two different imagesets of Douglas Fir (see Figure 1). The first one (**Besle**) consists of 65 (4320×3240 pixels) log end images of Douglas fir. The images were captured just after harvesting in forest with a digital camera (Panasonic Model DMC-FZ45). The second one (**BBF**) consists of 40 (4608×3456 pixels) log end images of Douglas fir. The images were captured with a smartphone (Huawei Model ANE-LX1). Both datasets concern the same log ends. As we were interested in ambient light variations, dirt or sawing marks, we captured images as raw as possible. There are 11 logs, but each log has been taken several times (from 3 to 5 times) by slightly rotating or moving the camera between each view. In average, a pixel stands for 0.191 mm, with a minimum at 0.111mm and a maximum at 0.1982 mm. In our study, we considered that pith position is a unique pixel even if from a biological point of view it may occupy a larger area. Ground truths, i.e. the pith center position, have been manually assessed for the evaluation.

5.2 Experiments

In our experiments, the ACO algorithm (Section 2) was processed twice. The first run was to coarsely estimate the pith while the second run was for a precise pith estimation. For the preprocessing, we split the image into 4×4 sub-images and preprocessed (see Section 2.1) each sub-image. Then, the pith detection is performed on the concatenation of these sub-images. We decided to proceed that way because sawing marks may not have the same orientation on the whole log end. And they may be less or more visible on some parts of the log end.

After retrieving the first pith estimation, the image was converted back to coordinates of the original image. We selected a sub-image centered on it and we run again the algorithm (including the preprocessing step but without subdivision). In practice, the sub-image for the second run was of size 512×512. Experiments were performed on an Intel Core i7 with 2,5 GHz and 16GB RAM.

To determine the best parameters' values, we minimized over the whole **BBF** dataset the sum of distances between ground truths and results. Then, we validated those values on the **Besle** dataset. Table 1 shows the best parameters obtained for the preprocessing.

	λ	δ	σ	**Band-pass**
For both runs	0.875	0.4	6	$\frac{H}{3} < f < \frac{H}{64}$

Table 1: Preprocessing parameters. H is the height of the Fourier spectrum.

Parameters for the ACO-based algorithms are shown in Table 1 and Table 2 (the values is the same for both runs unless otherwise indicated).

Parameter	K	α	β	γ	m	n	ω	κ	ε	N
Values	4	2.0	1.0	0.07	5 (then 2)	3	8	0.8	2 (then 0.5)	50

Table 2: ACO parameter values for experiments.

One of the most important parameters is δ, the reduction factor of the image. It should be mentioned that the proposed method relies on tree rings analysis, and the information between tree rings is less important for the detection. Due to the high resolution of both imagesets, this resizing step allows to reduce information between tree rings (without removing them) and also the computation time. We found that a value of 0.4 is right for our two imagesets.

The number of ants K^2 is crucial. It is a double-edged parameter, a large number of ants could increase the algorithm convergence and precision but it will increase computation time (see Section 3.2.1). Too few ants may lead to a wrong pith estimation. For images where the log end is large or the tree rings are clearly visible we may choose a lower number. But an image where the log end is small or the tree rings less visible would require a higher number of ants.

Two other parameters have been introduced to address our problem. The parameter m helps to reduce the computation time. We set a higher value of m for the first run as we wanted to have a fast and coarsely pith estimation and a lower value for the second run to improve accuracy. The other parameter is n. A low value of n would lead to fewer deposited pheromones and thus convergence would require more iterations. One would have to drastically increase the number of ants to make up for the lower value of n.

Tables 3 and 4 illustrate the method on a given image with the log end located far from the image center, in the top left quadrant of the image. For the first run, the algorithms early stopped at the 30^{th} iteration and few iterations were enough to have a good estimation. Around the 20^{th}, a region where the pith could be located can be observed. The second run takes 10 to 20 iterations to give stable results. The algorithms reached the maximum number of iterations.

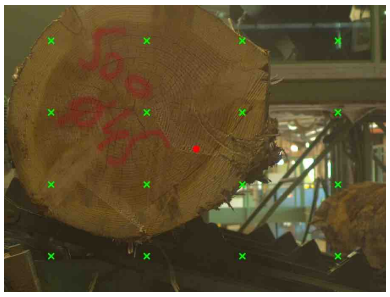
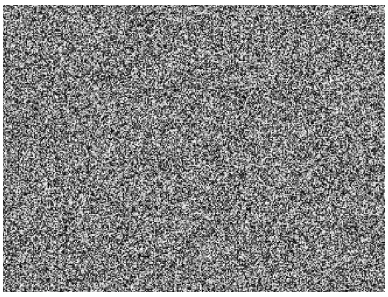
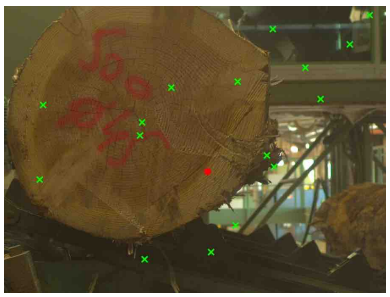
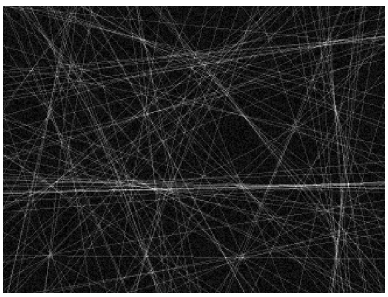
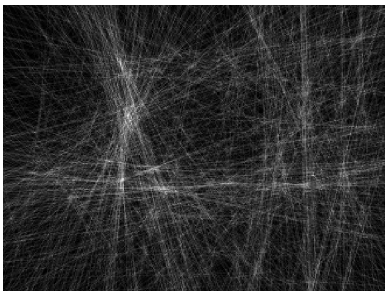
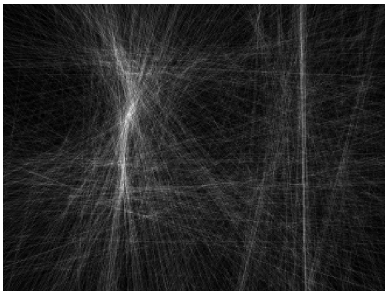
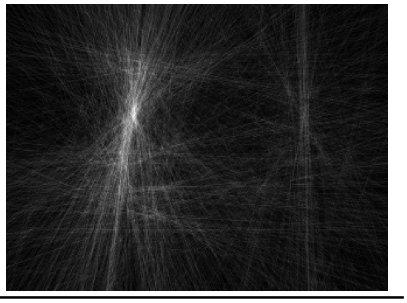
Iterations	Ants positions	Pheromone matrix τ
1		
2		
10		
20		
30		

Table 3: ACO algorithm for pith estimation. Results obtained at several iterations during the first run. In green cross ants positions, in red circle pith according to $\tau^{(t)}$. On the right column the pheromone matrix τ.

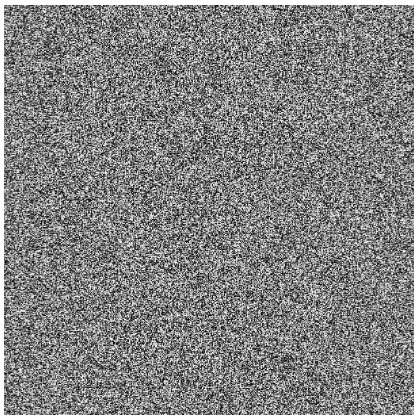
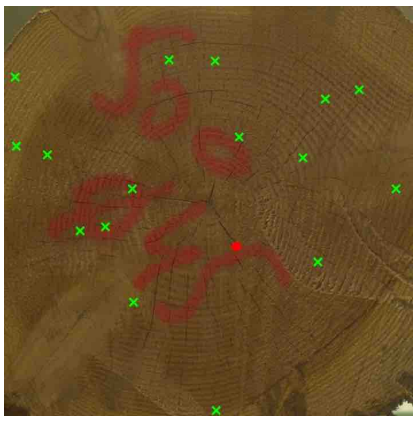
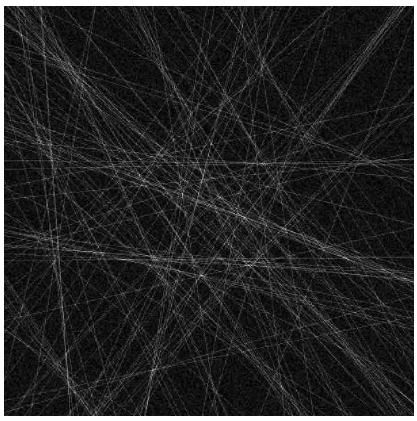
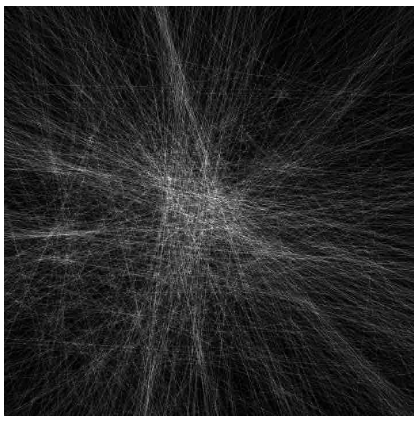
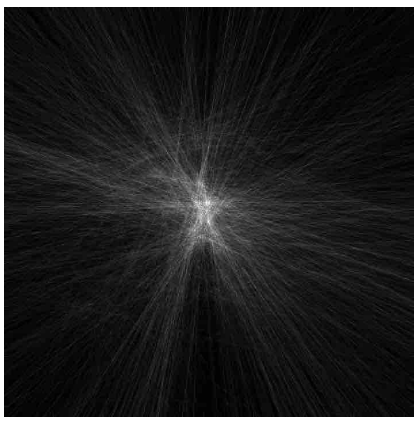
Iterations	Ants positions	Pheromone matrix τ
1		
2		
10		
50		

Table 4: ACO algorithm for pith estimation. Results obtained at several iterations during the second run. In green cross ants positions, in red circle pith according to $\tau^{(t)}$. On the right column the pheromone matrix τ.

5.3 Results

Although no method in the literature has been designed to operate in the experimental conditions described in the previous section, we have compared our results with [12],[17] and [11] on our datasets.

Kurdthongmee et al.,[12] divide the image into blocks and use Histogram of oriented Gradient (HoG) for the orientation estimate. Then, for each block, they accumulate a line according to the orientation with highest count of the histogram. Once the accumulation is finished, the pith is assumed to be the point with the highest accumulation.

However, depending on where the block is located in the image certain orientations cannot be selected. For instance, if the block is in the upper left, then orientations $\frac{\pi}{4}, \frac{5\pi}{3}$ cannot be accumulated.

We implemented the algorithm of Kurdthongmee et al.,[12] and computed optimized parameters with a subregion of size 24×24 pixels and a quantization factor of 12.

We also compared with Schraml and Uhl algorithm [17] using optimized parameters for our datasets. The method consists in dividing the image into blocks (which may overlap) and using Fourier analysis for the orientation estimate. Then it accumulates for each block a line according to the orientation estimate.

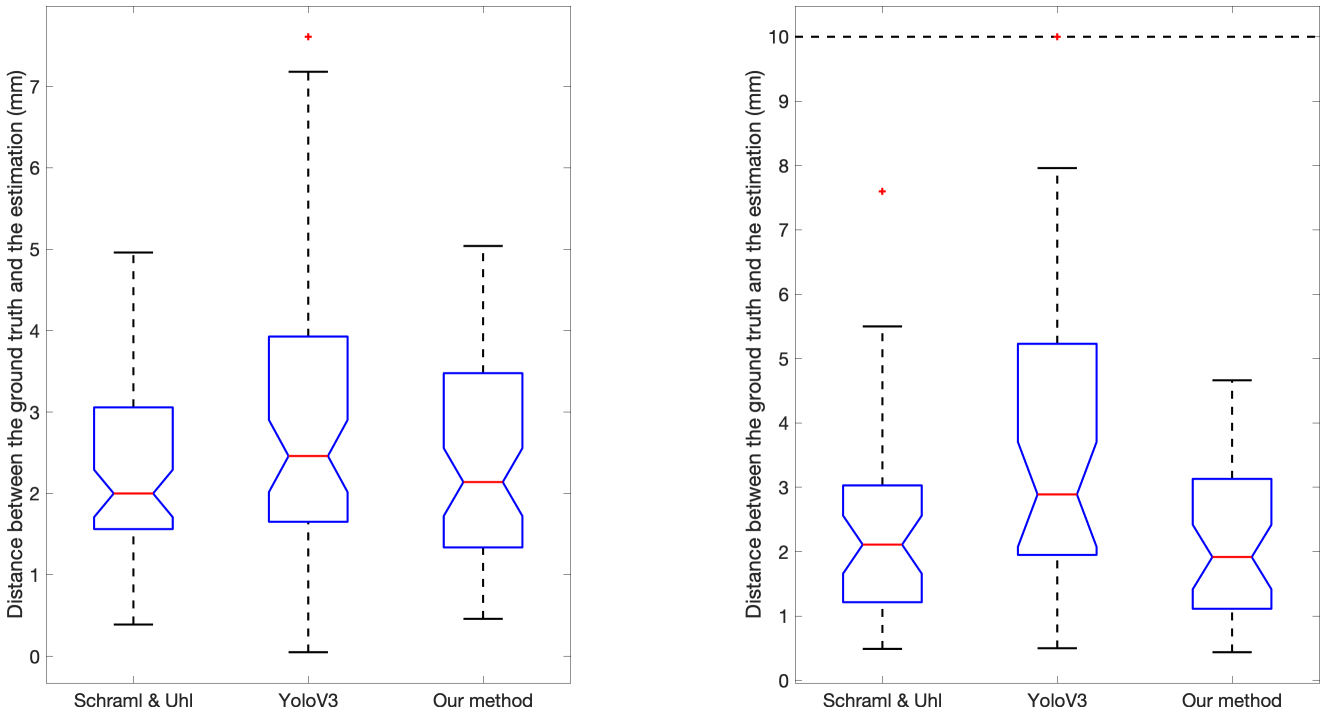
For the comparison with the DNN [11], we have performed a twofold cross-validation. For each imageset, half of the images have been used for the training and the other half for the validation. Two models were trained for each imageset by inverting the training and the validation sets. Ground truths consisted of a square of 300×300 pixels with the pith position at the center. As we have few images, data augmentation has been applied on-the-flight (i.e. each time each image was transformed before passing through the DNN). The transformations related to data augmentation are shift (vertical and horizontal), shear and zoom. The DNN hyperparameters are the same as in [11], only the input size has been modified, being 576×432 for both imagesets. The ratio is kept for each imageset and the ground truth is also resized according to that. The DNN returns a box with a probability of finding a pith on it. The predicted pith is the center of the box with the highest probability. To compare each method, we have aggregated all the predictions from the trained models (which gives us predictions for all images) and we have computed scores on this aggregation.

Table 5 presents a statistical analysis of the three algorithms on our datasets. The deep learning method is the fastest. The drawbacks are the learning time and the need to create huge datasets with ground truths. Our method is on average 5 times faster than [17] and can be easily parallelized. We can observe that both our method and [17] are more accurate than [11, 12]. The results from [11] are worse on the **BBF** imageset, this may be due to the small number of images in it.

Besle	Mean	StDev	Min	Max	Time (ms)
Schraml and Uhl [17]	**2.29**	**0.98**	**0.39**	**4.96**	8344
Kurdthongmee et al., [12]	25.06	21.23	2.15	92.44	667
YOLOv3 [11]	2.88	1.67	0.87	7.68	**138**
Our method	2.34	1.02	0.46	5.04	910
BBF	Mean	StDev	Min	Max	Time (ms)
Schraml and Uhl [17]	2.39	1.48	0.49	7.59	8660
Kurdthongmee et al., [12]	38.38	38.91	3.14	232.74	721
YOLOv3 [11]	12.69	53.55	0.50	341.92	**186**
Our method	**2.26**	**1.32**	**0.44**	**4.63**	1055

Table 5: Average, standard deviation, minimum and maximum between ground truths and estimated piths by our method and methods [17, 12, 11] (in mm) and average time to process one image (in ms).

Figure 8 presents boxplots for our method, [17] and [11] to better illustrate the differences between



(a) Boxplot for **Besle** dataset (b) Boxplot for **BBF** dataset

Figure 8: Boxplots of distance between ground truths and pith estimation (in mm) for Schraml and Uhl [17], the DNN (YoloV3) [11] and our method.

them. We excluded [12] since their results are less accurate than the three others. For the **Besle** imageset, our method is a little less accurate than [17].The DNN [11] is even a little less accurate and presents one outlier (7.61 mm). The first quartile and third quartile are higher than for our method and [17]. For the **BBF** imageset, [17] has one outlier (7.60 mm) and [11] has four outliers above 10 mm. But apart from them, DNN is close to both our results and the results of [17].

In Figure 9, two images of cross-sections are shown. The image on the left is the one where our method produces the highest error (5.04 mm). For this image the error of [11] is 7.16 mm and the error of [17] is 4.25 mm. The result of our method is acceptable even if it finds the pith slightly outside the first tree ring. The example on the right illustrates a mistaken detection by the DNN. This is due to the fact that the DNN may detect different piths on the image, and the final position reported could be on the wrong section. The error of [17] is 5.50 mm and our error is 4.21 mm. The parameter γ can help to remove pith estimations for small log ends. Indeed, small log ends would have less pheromones, then a higher value for γ will easily remove these unwanted pheromones.

In addition, we applied our algorithm on other images. Figure 10 shows these images and estimated piths. The method may fail if the tree rings are less visible (Figure 10a). We can observe that brightness does not modify pith estimations (Figure 10b–10c). Finally, our method works well on CT-images (Figure 10d).

6 Conclusion

In this paper, we proposed a method to estimate pith location on RGB images based on an ACO algorithm. The raw images were processed directly with no prior segmentation nor cropping. To the best of our knowledge, no method has been developed in such conditions. Despite this, we compared our method with other algorithms. Experiments were performed on two datasets of log end images

Figure 9: Two examples where the methods failed to be below 5mm from the ground truth. The ground truth is the white cross, results obtained by DNN [11] is the green one, Schraml and Uhl [17] is the blue one, and our method is the red one.

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

Figure 10: Examples of wood cross-section. Pith estimations by our method are indicated by red crosses.

of Douglas fir. The results showed a very accurate estimation. Our method is as accurate as other methods although we did not make any assumptions on the image. Furthermore, results showed a computational time low enough to be used in sawmill environment or in forest (with smartphones).

Processing other wood species in which tree rings are less visible is under investigation. For reducing computation time of the algorithm, parallelism should be considered in future work.

Acknowledgment

This research was made possible by support from the French National Research Agency, in the framework of the project TreeTrace, ANR-17-CE10-0016.

Image Credits

For Figure 10, the first image comes from [12]'s imageset. All the others images included in the article have been taken by the authors as part of the project TreeTrace.

References

[1] A.E. Akachuku and D.A.O. Abolarin, *Variations in pith eccentricity and ring width in teak (Tectona grandis LF)*, Trees, 3 (1989), pp. 111–116. https://doi.org/10.1007/BF01021074.

[2] H. Boukadida, F. Longuetaud, F. Colin, C. Freyburger, T. Constant, J.M. Leban, and F. Mothe, *PithExtract: A robust algorithm for pith detection in computer tomography images of wood–Application to 125 logs from 17 tree species*, Computers and electronics in agriculture, 85 (2012), pp. 90–98. http://dx.doi.org/10.1016/j.compag.2012.03.012.

[3] N. Dalal and B. Triggs, *Histograms of Oriented Gradients for Human Detection*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, IEEE, 2005, pp. 886–893. https://doi.org/10.1109/CVPR.2005.177.

[4] M. Dorigo, V. Maniezzo, and A. Colorni, *Ant system: optimization by a colony of cooperating agents*, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 26 (1996), pp. 29–41. https://doi.org/10.1109/3477.484436.

[5] R.O. Duda and P.E. Hart, *Use of the Hough transformation to detect lines and curves in pictures*, Communications of the ACM, 15 (1972), pp. 11–15. https://doi.org/10.1145/361237.361242.

[6] K. Entacher, D. Planitzer, and A. Uhl, *Towards an automated generation of tree ring profiles from CT-images*, in International Symposium on Image and Signal Processing and Analysis (ISPA), IEEE, 2007, pp. 174–179. http://dx.doi.org/10.1109/ISPA.2007.4383685.

[7] R. Gazo, J. Vanek, M. Abdul_Massih, and B. Benes, *A fast pith detection for computed tomography scanned hardwood logs*, Computers and Electronics in Agriculture, 170 (2020), pp. 105–107. https://doi.org/10.1016/j.compag.2019.105107.

[8] T. Hanning, R. Kickingereder, and D. Casasent, *Determining the average annual ring width on the front side of lumber*, in Optical Measurement Systems for Industrial Inspection III, vol. 5144, 2003, pp. 707–717. https://doi.org/10.1117/12.500200.

[9] L. Hong, Y. Wan, and A. Jain, *Fingerprint image enhancement: algorithm and performance evaluation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20 (1998), pp. 777–789. https://doi.org/10.1109/34.709565.

[10] M. Kass and A. Witkin, *Analyzing oriented patterns*, Computer Vision, Graphics, and Image Processing, 37 (1987), pp. 362–385. https://doi.org/10.1016/0734-189X(87)90043-0.

[11] W. Kurdthongmee, *A comparative study of the effectiveness of using popular DNN object detection algorithms for pith detection in cross-sectional images of parawood*, Heliyon, 6 (2020). https://doi.org/10.1016/j.heliyon.2020.e03480.

[12] W. Kurdthongmee, K. Suwannarat, P. Panyuen, and N. Sae-Ma, *A fast algorithm to approximate the pith location of rubberwood timber from a normal camera image*, in International Joint Conference on Computer Science and Software Engineering (JCSSE), IEEE, 2018, pp. 1–6. https://doi.org/10.1109/JCSSE.2018.8457375.

[13] F. Longuetaud, J-M. Leban, F. Mothe, E. Kerrien, and M-O. Berger, *Automatic detection of pith on CT images of spruce logs*, Computers and Electronics in Agriculture, 44 (2004), pp. 107–119. https://doi.org/10.1016/j.compag.2004.03.005.

[14] H. Nezamabadi-Pour, S. Saryazdi, and E. Rashedi, *Edge detection using ant algorithms*, Soft Computing, 10 (2006), pp. 623–628. https://doi.org/10.1007/s00500-005-0511-y.

[15] K. Norell, *Automatic counting of annual rings on Pinus sylvestris end faces in sawmill industry*, Computers and Electronics in Agriculture, 75 (2011), pp. 231–237. https://doi.org/10.1016/j.compag.2010.11.005.

[16] K. Norell and G. Borgefors, *Estimation of pith position in untreated log ends in sawmill environments*, Computers and Electronics in Agriculture, 63 (2008), pp. 155 – 167. https://doi.org/10.1016/j.compag.2008.02.006.

[17] R. Schraml and A. Uhl, *Pith estimation on rough log end images using local Fourier spectrum analysis*, in Conference on Computer Graphics and Imaging (CGIM'13), 2013. http://dx.doi.org/10.2316/P.2013.797-012.

[18] J. Tian, W. Yu, and S. Xie, *An ant colony optimization algorithm for image edge detection*, in IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 751–756. https://doi.org/10.1109/CEC.2008.4630880.

[19] Q. Wei, B. Leblon, and A. La Rocque, *On the use of X-ray computed tomography for determining wood properties: a review*, Canadian Journal of Forest Research, 41 (2011), pp. 2120–2140. http://dx.doi.org/10.1139/x11-111.