Pith Estimation on Tree Log End Images

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

[1] Université de Lorraine, CNRS, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, F-54506, France
[2] Université de Lorraine, AgroParisTech, INRAE, SILVA, F-54000 Nancy, France
remi.decelle@loria.fr

Abstract. In this paper, we present an algorithm for pith estimation from digital images of wood cross-sections. The method is based on a probabilistic approach, namely *ant colony optimization* (ACO). After introducing the approach, we describe the implementation and the reproduction of the method linking to an online demonstration. Results show that the approach performs as well as state-of-the-art methods. The estimated pith is below 5mm from the ground truth. It is a fast method that could be used in real-time environment. This paper also gives the details about the intern parameter choice and shows how to use the C++ source code for testing, as well as provides limit cases of the proposed method and future improvements.

Keywords: Agent-based method, Local orientation, Hough transform

1 Introduction

The centre of the annual rings, also called *pith*, is one of the most important feature to be detected since it can be related to wood quality [1,8,20] and it allows to extract other features on log-end image [6,7,10,17] such as annual rings, ring widths, knots, heartwood and sapwood. In the literature, several methods have been proposed for pith detection on log cross-sections. Most of them [2,3,9,14,15,24] have been developed for X-ray computed tomographic (CT) images. The techniques based on CT images allow an efficient and robust detection of external and internal characteristics of tree logs, including the pith. However, the CT scanners are very expensive, and not every laboratory or wood-industry sites (*e.g.,* sawmills) can acquire such a device.

Recently, there have been some efforts to develop pith detection methods on RGB images of cross sections from tree logs [12,13,19,21]. Contrary to CT images, RGB images exhibit disturbances like sawing marks, dirt or ambient light variations, which make the detection more challenging (see Fig. 1). On the other side, the acquisition of such images can be done with low-cost and more accessible devices (*e.g.,* smartphone camera, industrial camera, ...) and could be used everywhere, from the forest on the harvester, to the sawmill stocking area, or at the road side for wood sells. Furthermore, the current camera technologies

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

Fig. 1: Examples of image for pith detection: (a-b) CT images, (c-e) digital images captured in log-yard. Digital images taken in realistic environments may contain disturbances: (c) light condition, (d) soiling and dirt, (e) sawing marks.

provide images of quality and high resolution, and this allows us to extract the wood quality features, including pith, on such images using the image processing methods.

To the best of our knowledge, four works [12,13,19,21] have been published in the context of pith detection on digital images of rough, untreated log ends. Except in [12] which uses a deep neural network (DNN), the others rely on tree ring analysis and use image processing tools for the detection. More precisely, it is assumed that most tree rings close to the pith approximate a circular shape, and thus the normal directions of these rings would point towards the pith. Based on this idea, the pith detection is generally processed in three steps:

1. Estimate normal directions from tree ring local orientations.
2. Accumulate the normals in an accumulation space.
3. Extract the pith from the accumulation space.

Generally, Hough transform [5] is used as accumulator of normal directions, then the pith is identified at the point with maximum accumulation, or the barycenter of points above a given accumulation threshold. In other words, the pith detection methods differ at the local orientation estimation step. In [19], Norell and Borgefors presented two detection methods using two different techniques to estimate the normal directions: the quadrature filters and the Laplacian pyramids. The proposed methods are robust to disturbances; *e.g.,* rot, dirt or snow. However, a prior segmentation of the log end is needed before the detection. Later, Schraml and Uhl [21] proposed to compute the local orientations with Fourier spectrum analysis. The approach was fast, robust, and accurate in estimating the pith position. It, however, requires some preconditions about the cross-section size and its location in the image for initializing the computation. Recently, Kurdthongmee *et al.* [13] used histogram of oriented gradient (HOG) to estimate normals of the tree rings. As stated in the paper, the algorithm provides only an approximation of the pith location, and needs more treatments to identify it exactly.

In this paper, we propose a general method to estimate pith location on digital images taken in realistic environments; *e.g.,* in the sawmill, forest, log-yard, or on the road. The raw images are directly processed without any prior segmentation nor knowledge of log end visual appearance, shape or location. In

particular, the proposed method must not only provide accurate pith estimation, but also be efficient in computation time to be used in real-time applications. To this end, we consider a smooth gradient based method to compute local orientations; this method was originally used for fingerprint images [11]. Then, a probabilistic approach, based on Ant Colony Optimization (ACO) [4], is performed to accumulate the normals of tree rings in a robust way. Finally, the pith is located at the barycenter of points above a given accumulation threshold.

The proposed method is described in the following section (Section 2), with the details of the local orientation computation and the ACO algorithm. Section 3 gives the description of the source code and its usage. The experimental results and parameter discussions are addressed in Section 4, followed by the conclusion.

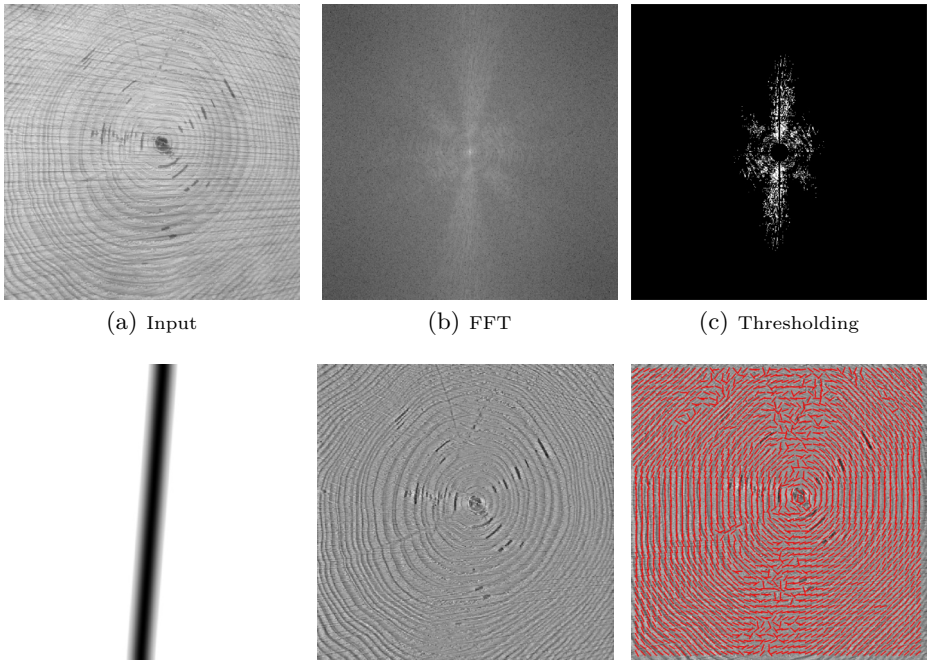
2 Algorithm for pith detection

The proposed algorithm to estimate pith location on digital images is composed of four steps. Firstly, a pre-processing is applied on input image to remove sawing marks visible on log ends. In case of high-resolution images, a resizing step can be applied to reduce the computation time. Secondly, we compute local orientations for pixels of the pre-processed image. Then, the ACO algorithm is used to accumulate the normals, and finally extract the pith from this accumulator. For an accurate pith estimation, the ACO algorithm and the pith extraction are performed twice: the first to coarsely estimate the pith region, and the second for the precise pith location.

2.1 Pre-processing image

Pith estimation methods based on ring analysis strongly depend on local orientation estimations. In this paper, we work with raw images in which there might be sawing marks on rough log ends. The presence of sawing marks perturbs the orientation estimations. To reduce errors induced by sawing marks and also computation time, we perform this pre-processing step. Firstly, the input image is converted into gray-scale, then down-sampled with a factor s using bi-linear interpolation. Secondly, we remove sawing marks using the method proposed in [18] which is based on Fast Fourier Transform (FFT).

Typically, sawing marks are straight lines being parallel or in fan-shape and not always evenly spaced. This repetitive pattern suggests that filtering in the Fourier domain is suitable to reduce them. Indeed, in the Fourier spectrum, they correspond to the line passing through the centre with a direction perpendicular to them. In other words, the energy level will be high along this line. Therefore, by reducing this energy and converting filtered spectrum back to spatial domain, sawing marks can be removed or at least reduced. More precisely, we first compute a Fast Fourier Transform (FFT) and filter the Fourier spectrum with a band-pass filter, also remove both horizontal and vertical lines. Then, we threshold it with a value λ to filter high value energy points corresponding to sawing marks. A line fitting, using principal component analysis (PCA), is



(a) Input (b) FFT (c) Thresholding

(d) Detected line (e) Sawing mark removal (f) Local orientation

Fig. 2: Removing sawing marks and estimation of local orientations: (a) input gray-scale image, (b) FFT of (a), (c) threshold (b) with $\lambda = 0.875$, (d) detected line after convoluted with a Gaussian of $\sigma = 6$, (e) image after removing sawing marks, and (f) estimated local orientations (red lines) on (e).

applied on the obtained points to find the direction of the line. This line is further convoluted with a Gaussian filter of standard deviation σ, and pixel-wise multiplied with the original Fourier spectrum to reduce energy along the line. The result is transformed back into a spatial image using the inverse FFT. The process is illustrated in Fig. 2 (a-e).

2.2 Local orientation

After removing sawing marks, we now compute normal directions for blocks of pixels in the image using a smooth gradient based method. The method was used in [11] to assess the local orientation in fingerprint images. It is a least mean square orientation estimation in a local area, namely a window of size w. More precisely, the gradient $\nabla(u, v) = [\nabla_x(u, v), \nabla_y(u, v)]^t$ is estimated for each

block centered at pixel (u, v) as follows.

$$\nabla_x(u,v) = \sum_{i=u-\frac{w}{2}}^{i=v+\frac{w}{2}} \sum_{j=u-\frac{w}{2}}^{j=v+\frac{w}{2}} 2\delta_x(i,j)\delta_y(i,j) \tag{1}$$

$$\nabla_y(u,v) = \sum_{i=u-\frac{w}{2}}^{i=v+\frac{w}{2}} \sum_{j=u-\frac{w}{2}}^{j=v+\frac{w}{2}} \delta_x^2(i,j)\delta_y^2(i,j) \tag{2}$$

where $\delta_x(i,j)$ and $\delta_y(i,j)$ are the derivatives with respect to x and y of the pixel (i,j). The derivatives are estimated by a Sobel operator [22]. Then, the local orientation of the block centered at (u,v) is computed as:

$$\theta(u,v) = \frac{1}{2}\tan^{-1}\left(\frac{\nabla_y(u,v)}{\nabla_x(u,v)}\right) \tag{3}$$

Fig. 2 (f) shows an example of local orientations estimated by this method.

2.3 Ant colony optimization

We now describe the process of accumulating normals using ACO which 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* [4] was the first ACO algorithm, it is applied for solving different combinatorial optimization problems; *e.g.,* traveling salesman problem (TSP), quadratic assignment problem (QAP) and the job-shop scheduling problem (JSP). Since then, a large number of ACO algorithms have been developed to address various problems like edge detection [16,23]. It has, to our knowledge, never been used as accumulator of local orientations.

The main idea of the proposed method is that a certain number of ants are uniformly placed on the rough log-end image. They can freely move on the image and use normal values as pheromones. Their final goal is the pith. Each ant iteratively lays down pheromones as it moves towards the pith, and they all move towards the pith area where there is a high quantity of pheromones. The process is summarized in **Alg. 1**. Hereafter, we describe in details the different steps of the proposed ACO algorithm.

Initialization (Line 1 and 2 in **Alg. 1**) Let $K \times K$ be the number of ants and $\pi^{(t)}$ be the pheromone matrix at iteration t. At the beginning, K ants are placed in an uniform grid on the pre-processed image, and the pheromone matrix $\pi^{(0)}$ is initialized with random values drawn from a normal distribution.

Computation of probabilistic transition matrix (Line 5 to 9 in **Alg. 1**) An ant can move randomly with a probability that evolves during the optimization.

Algorithm 1: Ant colony optimization for normal accumulation

Input: The estimated local orientations I_θ
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: The pheromone matrix π

Variables : η_k^t: The desirability matrix of the k^{th} ant at iteration t
τ_k^t: The probabilistic transition matrix of the k^{th} ant at iteration t
ρ_k^t: The deposited pheromone matrix of the k^{th} ant at iteration t
π^t: The pheromone matrix at iteration t

Initialize the positions of the K ants
Initialize the pheromone matrix π^0
for $t = 1 \dots N$ **do**
  for $k = 1 \dots K^2$ **do**
    Let (a, b) be the position of the k^{th} ant
    `/* Compute the deposited pheromone matrix` ρ^t `of the` k^{th} `ant */`
    Let B be the $n \times n$ block clusters of size $\omega \times \omega$ centered at (u, v) in I_θ
    foreach $h \in B$ **do**
      Let (o_x, o_y) be the median value of local orientations of h
      Let l be the line of orientation (o_x, o_y) passing through (u, v)
      Increase ρ_k^t by 1 along l
    `/* Compute the desirability matrix` η `of the ant to move towards a position` (u, v) `*/`
    foreach $(u, v) \in \eta_k^t$ **do**
      $\eta_k^t(u, v) = \frac{1}{\sqrt{(u-a)^2+(v-b)^2}+1}$
    `/* Compute the probabilistic transition matrix` ρ `of the ant to move towards a position` (u, v) `*/`
    foreach $(u, v) \in \tau_k^t$ **do**
      $\tau_k^t(u, v) = \frac{(\pi^t(u,v))^\alpha (\eta_k^t(u,v))^\beta}{\sum_{i,j} (\pi^t(i,j))^\alpha (\eta_k^t(i,j))^\beta}$
    `/* Move the ant according to the probabilistic transition matrix` τ_k^t `*/`
    Let (x, y) be the position of the maximum probability in τ_k^t
    Move the ant to the position (x, y)
  `/* Update the pheromone matrix` π^t `after all` K `ants moved */`
  $\pi^{t+1} = (1 - \gamma)\pi^t + \sum_{k=1}^{K} \rho_k^t$
  `/* Early-stopping criteria */`
  `/* Compute the new pith position */`
  The current pith position $\mathbf{p_{t+1}}$ is estimated according to π^{t+1}
  `/* Compute the distance between the current and the last pith position */`
  $d_{t+1} = \|\mathbf{p_{t+1}} - \mathbf{p_t}\|_2$
  `/* Compute the average of the last five distances */`
  $a_d = \frac{1}{5} \sum_{k=t-3}^{t+1} d_k$
  if $a_d < \varepsilon$ **then**
    Break
Return $\pi^{(N)}$

At iteration t, the probability for an ant k, currently at position (a, b), to move to the position (u, v) is defined by:

$$\tau_k^t(u,v) = \frac{\left(\pi^t(u,v)\right)^\alpha \left(\eta_k^t(u,v)\right)^\beta}{\sum_{i,j}\left(\pi^t(i,j)\right)^\alpha \left(\eta_k^t(i,j)\right)^\beta} \tag{4}$$

where $\tau^t(u, v)$ is the amount of pheromone at (u, v), $\eta_k^t(u, v)$ is the desirability of the k^{th} ant to move towards (u, v), and equal to the inverse distance from (u, v) to (a, b):

$$\eta_k^t(u,v) = \frac{1}{\sqrt{(u-a)^2 + (v-b)^2} + 1} \tag{5}$$

The desirability can be seen as a weighting of pheromone matrix. It aims to ensure ants having a higher probability to move towards local maxima and not towards the global one. α and β are respectively parameters to control the influence of $\tau_k^t(u, v)$ and $\eta_k^t(u, v)$. The ratio $\frac{\alpha}{\beta}$ allows to modify the behavior of ants; a high ratio value leads ants to move more quickly to the pheromone peaks, while a low value leads ants to continue to explore areas in image.

Pheromone deposit (Line 12 to 16 in **Alg. 1**) Each ant is the centre of an image block cluster. The cluster consists of $n \times n$ blocks, and each block has a size of $\omega \times \omega$ pixels. For each block cluster, the median value of local orientations (see Section 2.2) is considered as the block orientation. Then a line is drawn according to the orientation and passing through (u, v). All elements of the deposited pheromone matrix ρ_k^t along the line are incremented by 1. Indeed, depositing pheromones along the whole line allows to include lines intersections which could not happened if pheromone deposit is locally done. Fig. 3 illustrates this step of pheromone deposit of an ant.

Updating the pheromone matrix (Line 17 in **Alg. 1**) The pheromone matrix is updated once all ants have moved:

$$\pi^{t+1} = (1-\gamma)\pi^t + \sum_{k=1}^{K} \rho_k^t \tag{6}$$

where ρ_k^t is the deposited pheromone matrix of the k^{th} ant at iteration t, and γ is the rate of pheromone evaporation; the higher γ is, the faster pheromones are removed.

The process is repeated maximum N times (Line 3 in **Alg. 1**). In order to reduce computational time, the pheromone matrix π^t is resized by a factor m comparing to the pre-processed image I. In other words, if I is of size $H \times W$, then π^t is of size $\frac{H}{m} \times \frac{W}{m}$.

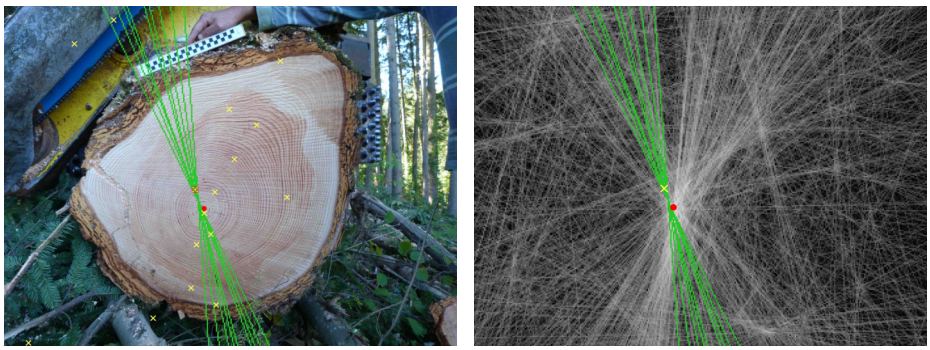


Fig. 3: Left: Image with the ants in yellow crosses, the pith in red, a cluster of 3×3 blocks of size 8×8 pixels and normals (according to the cluster) in green lines. Right: Normal accumulation by pheromone matrix with the considered ant in yellow cross, the 9 green lines corresponding to the normal directions of the block clusters and the pith estimation in red.

2.4 Pith extraction

To extract the pith position from the pheromone matrix π, we take the barycentre of all the pixels above $\kappa * \max(\tau)$ in π. Indeed, taking the maximum value of accumulation is less robust than the barycentre of the highest values. Note that the higher κ is, the more sensitive to small variations the pith estimation is.

Furthermore, we introduce an early stopping criterion in **Alg. 1**. At each iteration t, we estimate the pith location and compute the distance between the current and the last estimation. Instead of running for N iterations, the algorithm could stop as soon as the average of the last five distances falls below a threshold ε.

3 Code sources

3.1 Download and installation

The proposed method is implemented in Matlab 2019b and C++ using the open source library OpenCV[3] (OPEN Computer Vision). Both implementations are available at the github repository:

`https://gitlab.com/Ryukhaan/treetrace/-/tree/master/pith`

The installation is done with a cmake[4] procedure (see *README.md*[5]). In the following, we focus on the C++ implementation.

[3] `https://opencv.org/`
[4] `http://www.cmake.org`
[5] `https://gitlab.com/Ryukhaan/treetrace/-/blob/master/README.md`

3.2 Description and usage

The repository has four packages:

- **aco** computes the Ant Colony Optimization algorithm for one image;
- **normals** computes normal accumulations using Bresenham lines;
- **orientation** computes local orientations for one image;
- **ui** manages the display (pheromones, ants position on image, and so on).

Once the installation is done, the executable file is in the **build** directory and named **AntColonyPith**.

- **Input:** The image to be processed;
- **Command Line:** To run the program from the CODESOURCES/build

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

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

For instance, to run the program on **harvest.jpeg** with default parameters

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

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

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

or

```
./AntColonyPith --input ../../samples/harvest.jpeg -n 10 -a 1.0 --animated=false
```

To run the program on **harvest.jpeg** with parameters in **parameters.json**

```
./AntColonyPith --input ../../samples/harvest.jpeg
                --parameters ../AntColonyPith/parameters.json
```

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

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

The options can be provided in two ways:
  - using command line with usual options,
  - providing a JSON file with all parameters (an example of JSON file, namely *parameters.json*[6], is provided within the repository).
- **Output:** Two files are created. The first one consists of the detected pith position in CSV format. The second one is an image of the input image with the detected pith denoted by a cross.

[6] `https://gitlab.com/Ryukhaan/treetrace/-/blob/master/pith/c++/AntColonyPith/parameters.json`

Fig. 4: Examples from **Besle** (the first two rows) and **BBF** (the last row) datasets.

4 Experimental results

4.1 Experiments on real images

We experiment our algorithm on two datasets obtained from Douglas fir trees: **Besle** consists of 65 images and **BBF** consists of 40 images (see Fig. 4 for some examples). RGB images are converting into grayscale using the usual weighted method. Both datasets include the raw log ends taken in the forest or log yard, the images contain different disturbances such as sawing marks, dirt and light variations. Some visual results are shown in Fig. 5 by running the proposed method with the default parameters. The values of default parameters are given in the file *parameters.json*[6]. Further experiments about computation time and algorithm convergence are presented in the next sections. Note that we take the average value overall experiments on images in both datasets.

4.2 Accuracy of the method

For our implementation, we process twice the described method Section 2. RBG image are converted into grayscale (with the function *imread* and the option *IMREAD_GRAYSCALE* from Opencv). The first run is to coarsely estimate the pith while the second run is for a precise pith estimation. For the first run, we split the image into 4×4 sub-images to manage the sawing marks removal (see Section 2.1). After retrieving the first pith estimation, this latter is converted back to coordinate of the original image. We select a sub-image of size 512×512 pixels centered on it and process again the algorithm (including the preprocessing without subdivision).

Ground truths were done by two operators. Each operator independently, for each image, pointed the pith. The truth is the average of these two measures.

To determine the parameters' values, we manually minimized over the whole **BBF** dataset the sum of distances between ground truths and results. Then,

Fig. 5: Pith position (black cross) detected by the proposed method on raw log-end images using default parameters.

Table 1: Pre-processing parameters for both steps. H is the height of the Fourier spectrum.

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

we validated those values on **Besle** dataset. Tables 1 shows the parameters obtained for the preprocessing. Parameters for the ACO-based algorithms are set as follows (the values are the same for both phase unless otherwise indicated):

- $K = 16$: the number of ants $K \times K$;
- $\alpha = 2.0$: the control of the pheromone influence in (4);
- $\beta = 1.0$: the control of the heuristic influence in (4);
- $\gamma = 0.07$: the evaporation rate in (6);
- $m = 5$ for the first run then $m = 2$ for the second one: how many pixels an element in the matrix τ stands for;
- $n = 3$: the size of the blocks cluster (see Section 2.3);
- $\omega = 8$: the size in pixels of a block (see Section 2.3);
- $\kappa = 0.8$: threshold to the barycentre (see Section 2.4)
- $\varepsilon = 2$ for the first run then $\varepsilon = 0.5$: the thresh to early stop the algorithm.
- $N = 50$: the maximum number of iterations;

We have compared our results with [12], [13] and [21] on our datasets. For the algorithm of Kurdthongmee et al.,[13], we get optimized parameters with a

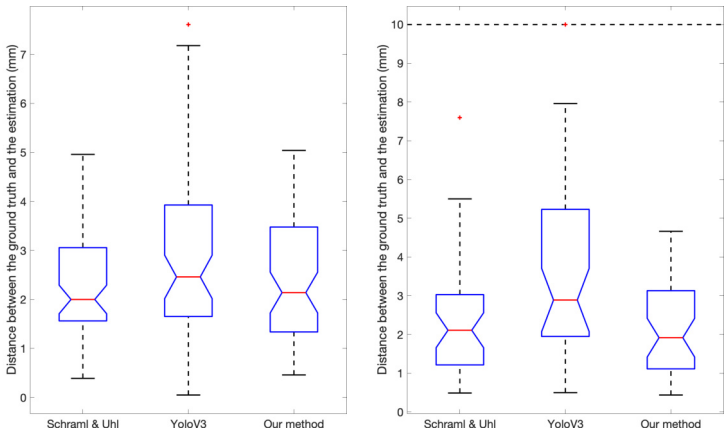
Table 2: Average, standard deviation, minimum and maximum between ground truths and estimated piths by our method and methods of [21,13,12] methods (in mm) and average time to proceed one image (in ms).

Besle	Mean	StDev	Min	Max	Time (ms)
[21]	**2.29**	**0.98**	**0.39**	**4.96**	8344
[13]	25.06	21.23	2.15	92.44	667
[12]	2.88	1.67	0.87	7.61	**138**
Our method	2.34	1.02	0.46	5.04	1611
BBF	Mean	StDev	Min	Max	Time (ms)
[21]	2.39	1.48	0.49	7.59	8660
[13]	38.38	38.91	3.14	232.74	721
[12]	12.69	53.55	0.50	341.92	**186**
Our method	**2.26**	**1.32**	**0.44**	**4.63**	1745

subregion of size 24×24 pixels and a quantization factor of 12. We also used optimized parameters of our dataset for Schraml and Uhl algorithm [21]. For the comparison with the DNN [12], we have done a twofold cross-validation. For each imageset, half of 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 consist of a square of 300×300 with the pith position at the center. A data augmentation have been processed on-the-flight (i.e. each time each image was transformed before passing through the DNN). The DNN hyperparameters were the same as [12], only input size have been modified which is 576×432 for both imageset (the ground truth is resized according to that). The DNN returns a box with a probability of finding a pith in it. The predicted pith is the center of the box with the highest probability. To compare each method, we have aggregated all predictions from trained models (which gives us predictions for all images).

Table 2 presents a statistical analysis of the three algorithms on our datasets. The deep learning method is the fastest but drawbacks are the learning time and the creation of dataset with ground truths. Our method is, in average, 5 times faster than [21] and can be easily parallelized. We can observe that both our method and [21] are more accurate than [12,13]. The results [12] are worse on **BBF** imageset, this may be due to the small number of images in it.

Fig. 6 presents boxplots for our method, [21] and [12] to better illustrate the differences between them. We excluded [13] since the results are less accurate than the three others. For **Besle** imageset, our method is a little less accurate than [21]. [12] is even a little less accurate and presents one outlier (7.61 mm). Its first and third quartiles are higher than our method and [21]. For **BBF** imageset, [21] has one outlier (7.6 mm) and [12] has four outliers above 10 mm.



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

Fig. 6: Boxplots of distance between ground truths and pith estimation (in mm) for [21,12] and our method.

4.3 Effect of parameter changes on computation time

Hereafter, the experiments are carried out to analyse the effects of parameter changes. First, we focus on computation time then on precision and convergence.

Let estimate the time complexity for one iteration in **Alg. 1**. Let I be the input image of size $H \times W$. According to **Alg. 1**, there are K^2 ants, and they can freely move on I. Let now estimate the time complexity for an ant, namely the k^{th} ant. Firstly, the desirability matrix η and the probabilistic transition matrix ρ associated to the k^{th} ant are computed in $\mathcal{O}(\frac{HW}{m^2})$. Secondly, for the pheromone deposit, we must recall that each ant is the centre of an image block cluster which consists of $n \times n$ blocks and each block is of size $\omega \times \omega$. Therefore, to estimate the block orientation, we compute n^2 times the median of an array of ω^2 pixels. This operation is done by sorting the array and costs $\mathcal{O}(n^2\omega^2 \log \omega)$. We also compute n^2 times the deposited pheromone matrix ρ_k^t along the directional line l. In the worst case, the length of l is equal to $\sqrt{(\frac{H}{m})^2 + (\frac{W}{m})^2}$. In other words, the pheromone matrix update is done in $\mathcal{O}(\frac{n^2}{m}\sqrt{H^2 + W^2})$. Finally, the ant's position is updated in $\mathcal{O}(1)$. Once each ant has moved, the pheromones matrix π^t is updated in $\mathcal{O}(\frac{HW}{m^2})$. Therefore, the total time complexity for one iteration is:

$$\mathcal{O}\left(K^2\left[n^2\omega^2 \log \omega + \frac{n^2}{m}\sqrt{H^2 + W^2} + \frac{1}{m^2}HW + 1\right] + \frac{1}{m^2}HW\right) \tag{7}$$

We can simplify this equation by keeping only the main input parameters, which are K, H and W. The total time complexity is therefore:

$$\mathcal{O}\left(K^2\left[\sqrt{H^2 + W^2} + HW + 1\right]\right) \tag{8}$$

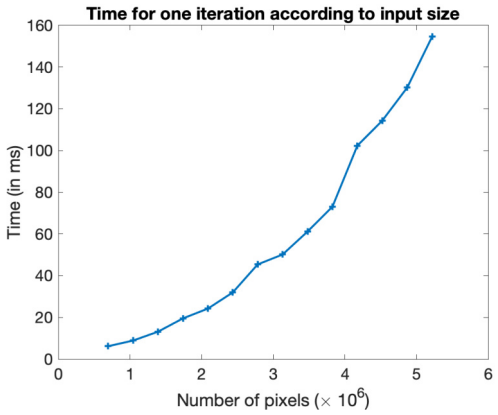


Fig. 7: Computation time for one iteration according to the size of input image (in ms).

We now validate this theoretical time complexity by the experiments. Firstly, it can be seen in *Eq.* (7) that the method is quadratic with respect to the size of the input image I. This is confirmed by Fig. 7, we have the computation time for one iteration according to I.

From *Eq.* (7), the method has a linear time complexity with respect to the number of ants K^2. Indeed, Fig. 8 (a) shows the computation time according to K. The number of ants quadratically increases, and thus the computation time.

Still in *Eq.* (7), the computation time decreases as m increases. Fig. 8 (b) shows the computation time according to m. It can be seen that having a value of m higher than 1 is really computationally helpful. Indeed, the matrix π is widely used during the process, from the pheromones deposit to the pheromones updates (reducing the size of η, τ and π). Let now look at two parameters: the block size $\omega \times \omega$ and the number of clusters $n \times n$ around the ant. First, the computation time according to the block size is shown in Fig. 8 (c). As the number of block is at least one, the computation time does not start at 0. For a block size of 3×3 it takes in average 243ms, while for a block size of 11×11 it is 259ms. Note that the computation time depends on the length of the line l used to update ρ (which depends on local orientations). For n, it should be quadratic. As n should be at least one, the computation time does not start at 0. Fig. 8 (d) shows the computation time according to n, and it is nearly quadratic. Again, this is due to the length of l.

4.4 Effects of parameter changes on the convergence

We now analyse the influence of parameters regarding the convergence of the algorithm. More precisely, the algorithm converges if the Euclidean distance between two successive estimations does not vary more than one pixel. We compute the average variations at each iteration over the whole set of images. We also fit a curve $ae^{bx} + c$ for each change in the value of parameters. It is done by

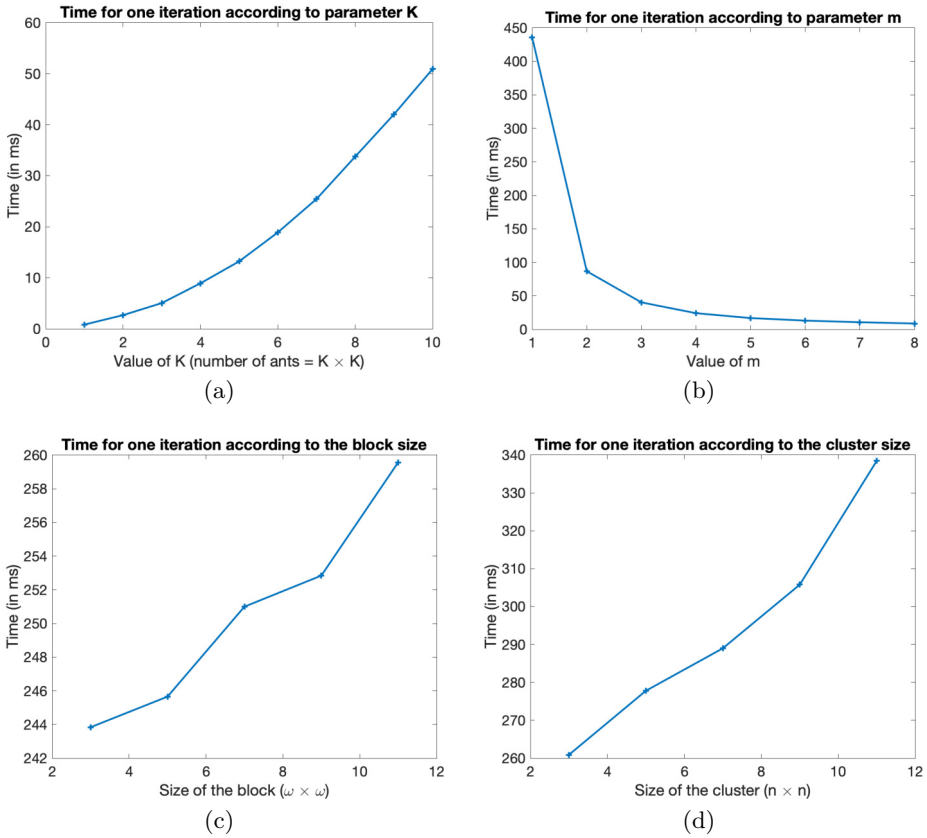


Fig. 8: Computation time for one iteration according to (a) the number of ants K, (b) the size of τ, (c) the block size ω and (d) the cluster size n.

using non linear least squares with trust-region algorithm and for initial value $a = \frac{\max - \min}{2}, b = 0$ and $c = 0$.

First, we focus on the number of ants $K \times K$. Fig. 9 shows the convergence speed according to K. One can see that the more ants are, the fastest the algorithm converges. With only four ants, the algorithm fluctuates between some positions. However, a high number of ants does not speed up convergence but makes the convergence point more stable (the parameter c is lower with a high value of K).

Let us now focus on the block's size ω. Fig. 11 shows the convergence speed according to ω. It seems that large block speeds up convergence but not as sharply as K. It is observed that an increasing in ω seems to slow down the convergence. This is due to an increase in the deposited pheromones. Indeed, the higher ω is, there more pheromones in the *wrong* places are. Therefore, it requires more iterations to remove those pheromones.

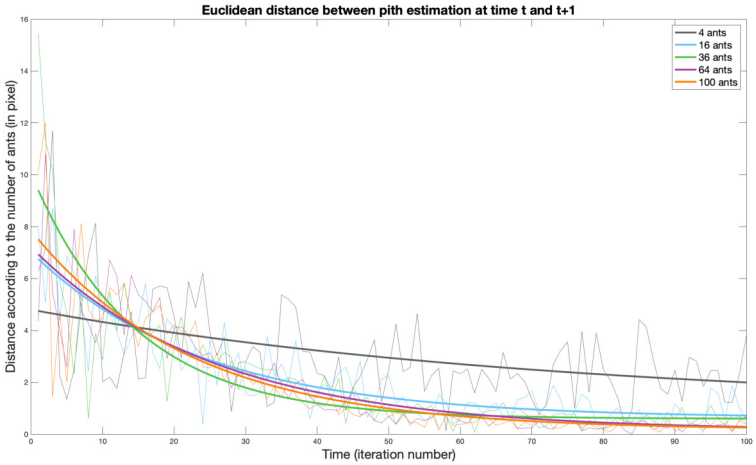


Fig. 9: Variation in both axis between the pith estimation at time t and at time $t+1$ according to the number $K \times K$ of ants.

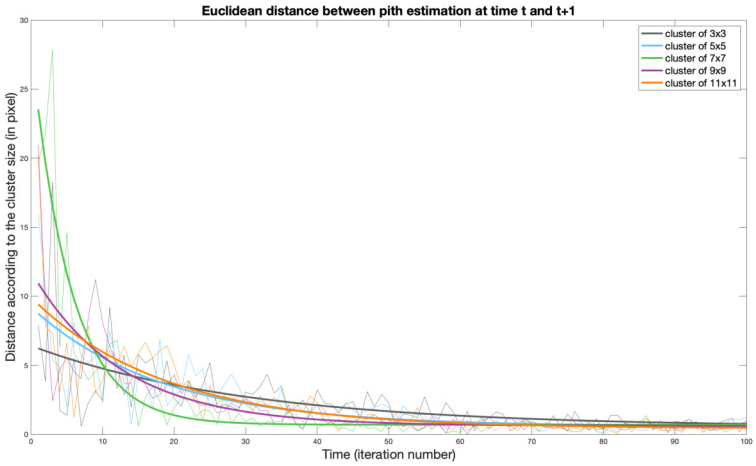


Fig. 10: Variation in both axis between the pith estimation at time t and at time $t+1$ according to the cluster size ω.

Let us now consider the number of blocks n. Fig. 10 shows the convergence speed according to n. Contrary to intuition, a large number of blocks does not lead to an important acceleration of convergence, but it slows the algorithm down. This could be due to a less accurate local orientation with larger blocks.

Finally, we look at m. As a reminder, the higher m is, the smaller τ is. Fig. 12 shows the convergence speed according to m. It appears that m slightly speeds up the convergence. Raising the value of m causes a slightly decelera-

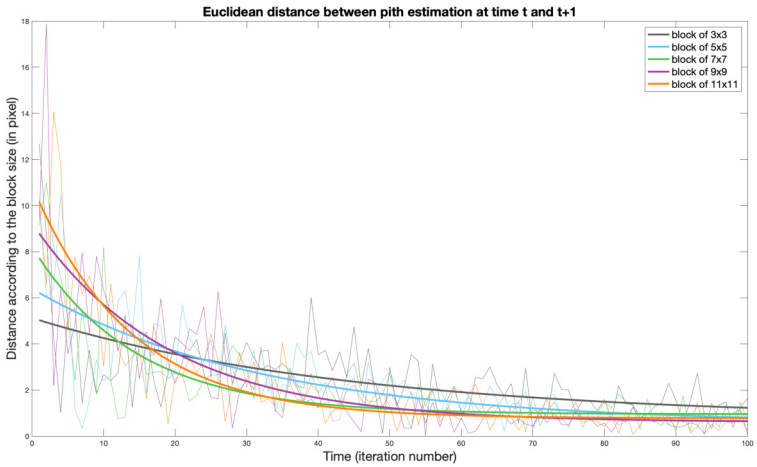


Fig. 11: Variation for both axis between the pith estimation at time t and at time $t+1$ according to the block size n.

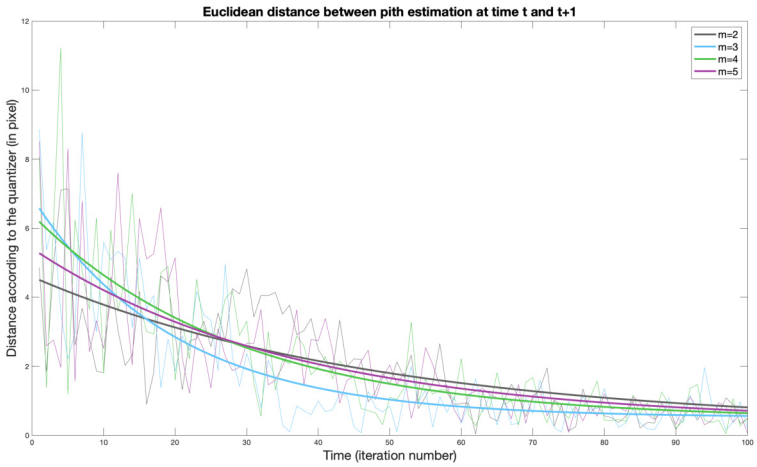


Fig. 12: Variation in both axis between the pith estimation at time t and at time $t+1$ according to the parameter m.

tion the convergence. It could be explained by the fact that with a small τ the small variations in the local orientation are not considered when ants deposits pheromones.

4.5 Limit cases

Our algorithm relies mainly on tree ring analysis. In other words, if the tree rings are not well presented in the input log-end image, then the detection result

Fig. 13: Examples of wrong pith estimation. Left: The tree rings are in low resolution and the sawing marks are not straight lines. Right: the sawing marks are straight lines but tree rings are in low resolution and really small.

could be inaccurate. Fig. 13 shows some examples in which tree ring analysis is difficult and leads to a wrong pith estimation by using the proposed method. These two examples are not from both datasets. Indeed, our algorithm works very well on introduced datasets (no outliers). In both examples, it can be observed that tree rings are barely visible that makes difficult their analysis. Indeed, the pre-processing step may remove information about tree rings, which leads our algorithm to an inaccurate pith estimation. Furthermore, there are many other disturbances on such images; for instance, the sawing marks are not straight lines as we assumed in Section 2.1, or the presence of log-tree cracks.

4.6 Image credits

All images used in this paper are from the French National Research Agency, in the framework of the project TreeTrace, ANR-17-CE10-0016. Some samples (images in Fig. 5 and Fig. 13) are available for testing on the github repository[7].

5 Conclusion

In this paper, we presented a probabilistic method for detecting pith position on digital images of rough, untreated log ends. More precisely, the proposed method is based on ant colony optimization (ACO) to robustly accumulate the normals of ring tree, then the pith location is extracted from this accumulation space as barycenter of points above a threshold. The experiments demonstrated that the proposed method provides not only an accuracy pith estimation (a distance of

[7] `https://gitlab.com/Ryukhaan/treetrace/-/tree/master/pith/samples`

less than 5 mm from the ground truths), but also is efficient in computation time and it could be used in real-time applications. In addition to the implementation of the method, an online demonstration is available for testing at:

`https://ngophuc.github.io/ACO_PithDetection_IPOLDemo`

It could be noticed that the algorithm has many parameters. Though, they are set with default values and allow a good performance on tested images. Generally, the algorithm provides a very accurate pith estimation. A study on the role and effect of the different parameters is addressed in the paper for a better understanding of the parameters on the presented method. Based on this study, a perspective is to reduce the number of parameters and further provide an automatic approach to determine the best parameters adapted to a given image or a set of images of same characteristic. Furthermore, 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.

References

1. Akachuku, A., Abolarin, D.: Variations in pith eccentricity and ring width in teak (tectona grandis lf). Trees 3(2), 111–116 (1989)
2. Bhandarkar, S.M., Faust, T.D., Tang, M.: A system for detection of internal log defects by computer analysis of axial ct images. In: Proceedings of the 3rd IEEE Workshop on Applications of Computer Vision (WACV '96). p. 258. IEEE Computer Society, USA (1996)
3. Boukadida, H., Longuetaud, F., Colin, F., Freyburger, C., Constant, T., Leban, J., Mothe, F.: 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, 90–98 (2012)
4. Dorigo, M., Maniezzo, V., Colorni, A., et al.: Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, man, and cybernetics, Part B: Cybernetics 26(1), 29–41 (1996)
5. Duda, R.O., Hart, P.E.: Use of the hough transformation to detect lines and curves in pictures. Communications of the ACM 15(1), 11–15 (1972)
6. Entacher, K., Planitzer, D., Uhl, A.: Towards an automated generation of tree ring profiles from ct-images. In: Image and Signal Processing and Analysis (ISPA). 5th International Symposium on. pp. 174–179. IEEE (2007)
7. Fabijanska, A., Danek, M., Barniak, J., Piórkowski, A.: Towards automatic tree rings detection in images of scanned wood samples. Computers and Electronics in Agriculture 140, 279–289 (2017)
8. Fallah, A., Riahifar, N., Barari, K., Parsakhoo, A.: Investigating the out-of-roundness and pith-off-centre in stems of three broadleaved species in hyrcanian forests. Journal of Forensic Sciences 58, 513–518 (2012), `https://doi.org/10.17221/13/2012-JFS`

9. Gazo, R., Vanek, J., Abdul_Massih, M., Benes, B.: A fast pith detection for computed tomography scanned hardwood logs. Computers and Electronics in Agriculture 170, 105–107 (2020)
10. Hanning, T., Kickingereder, R., Casasent, D.: Determining the average annual ring width on the front side of lumber. In: Optical Measurement Systems for Industrial Inspection III. vol. 5144, pp. 707–717 (2003)
11. Hong, L., Wan, Y., Jain, A.: Fingerprint image enhancement: algorithm and performance evaluation. IEEE transactions on pattern analysis and machine intelligence 20(8), 777–789 (1998)
12. Kurdthongmee, W.: A comparative study of the effectiveness of using popular dnn object detection algorithms for pith detection in cross-sectional images of parawood. Heliyon 6(2) (2020)
13. Kurdthongmee, W., Suwannarat, K., Panyuen, P., Sae-Ma, N.: A fast algorithm to approximate the pith location of rubberwood timber from a normal camera image. In: 15th International Joint Conference on Computer Science and Software Engineering (JCSSE). pp. 1–6. IEEE (2018)
14. Longuetaud, F., Mothe, F., Kerautret, B., Krähenbühl, A., Hory, L., Leban, J.M., Debled-Rennesson, I.: Automatic knot detection and measurements from x-ray ct images of wood: A review and validation of an improved algorithm on softwood samples. Comput. Electron. Agric. 85, 77–89 (2012)
15. Longuetaud, F., Leban, J.M., Mothe, F., Kerrien, E., Berger, M.O.: Automatic detection of pith on ct images of spruce logs. Computers and Electronics in Agriculture 44(2), 107–119 (2004)
16. Nezamabadi-Pour, H., Saryazdi, S., Rashedi, E.: Edge detection using ant algorithms. Soft Computing 10(7), 623–628 (2006)
17. Nordmark, U.: Models of knots and log geometry of young pinus sylvestris sawlogs extracted from computed tomographic images. Scandinavian journal of forest research. 18(2), 168—175 (2003), https://doi.org/10.1080/02827580310003740
18. Norell, K.: Automatic counting of annual rings on Pinus sylvestris end faces in sawmill industry. Computers and Electronics in Agriculture 75(2), 231–237 (Feb 2011)
19. Norell, K., Borgefors, G.: Estimation of pith position in untreated log ends in sawmill environments. Computers and Electronics in Agriculture 63(2), 155 – 167 (2008)
20. Rune, G., Warensjo, M.: Basal sweep and compression wood in young scots pine trees. Scandinavian journal of forest research. 17(6), 529—537 (2002), https://doi.org/10.1080/02827580260417189
21. Schraml, R., Uhl, A.: Pith estimation on rough log end images using local fourier spectrum analysis. In: Proceedings of the 14th Conference on Computer Graphics and Imaging (CGIM'13), Innsbruck, AUT (2013)
22. Sobel, I., Feldman, G.: An isotropic 3x3 image gradient operator. In: History and Definition of the so-called "Sobel Operator" (1990)
23. Tian, J., Yu, W., Xie, S.: An ant colony optimization algorithm for image edge detection. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). pp. 751–756. IEEE (2008)
24. Wei, Q., Leblon, B., La Rocque, A.: On the use of x-ray computed tomography for determining wood properties: a review. Canadian journal of forest research 41(11), 2120–2140 (2011)