Automatic Analysis of Wood Growth in the Sawmill Industry Using Deep Learning

Jukka Laakko

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 23.12.2019

Supervisor

Prof. Jukka Suomela

Advisor

Lic.Sc. (Tech.) Jari Mononen

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Jukka Laakko	
Title Automatic Analysis of Wood Growth in the Sawmill Industry Using Deep Learning	
Degree programme Computer, Communication and Information Sciences	
Major Computer Science	**Code of major** SCI3042
Supervisor Prof. Jukka Suomela	
Advisor Lic.Sc. (Tech.) Jari Mononen	
Date 23.12.2019	**Number of pages** 70+5 **Language** English

Abstract
Analysis of wood growth is an important quality control step in a sawmill, as it predicts the structure and load-bearing capabilities of the wood. The annual growth of wood is determined by calculating the distances between the annual rings in a wood end-face. The wood is moving fast in a process line, and manual analysis of wood growth is a laborious task that is prone to errors. Having the process automated increases the efficiency and throughput of the sawmill as well as reduces monotonic manual labor, thus providing better working conditions.

Automatic counting of annual ring distances has been studied before, however, little research has been done on a sawmill setting which suffers from difficult imaging conditions and rough wood end-faces with various defects. Previous studies have used traditional image processing methods which rely on handcrafted features and fail to generalize well on wood end-faces with varying conditions and arbitrary shaped annual rings.

This thesis proposes a general solution to the problem by developing complete end-to-end software for detecting annual rings and analyzing wood growth using deep learning methods. The proposed system is described in detail and compared against traditional computer vision methods. Using data from a real sawmill, the deep learning based approach performs better than the traditional methods.

Keywords Computer Vision, Deep Learning, Convolutional Neural Networks, Semantic Segmentation, Sawmill, Annual Rings

Aalto University
School of Science

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Jukka Laakko	
Työn nimi Automaattinen syväoppimiseen perustuva puun vuosikasvun analysointi sahateollisuudessa	
Koulutusohjelma Computer, Communication and Information Sciences	
Pääaine Computer Science	**Pääaineen koodi** SCI3042
Työn valvoja Prof. Jukka Suomela	
Työn ohjaaja TkL Jari Mononen	
Päivämäärä 23.12.2019	**Sivumäärä** 70+5 **Kieli** Englanti

Tiivistelmä

Puun vuosikasvun analysointi on tärkeä osa laadunvarmistusta sahalla, sillä vuosikasvu määrittää puun rakenteen ja kestävyyden. Lankut kulkevat nopeasti tehdaslinjastolla, joten manuaalinen vuosikasvun analysointi on vaivalloista ja virhealtista työtä. Prosessin automatisointi lisää sahan suoritustehoa sekä vapauttaa työntekijän mielekkäämpiin tehtäviin.

Puun vuosikasvu määritetään selvittämällä vuosirenkaiden väliset etäisyydet lankun päädystä. Automaattista vuosirenkaiden laskentaa on käsitelty kirjallisuudessa aiemmin, mutta vain muutama tutkimus on tehty sahaympäristössä, jossa kuvausolosuhteet ovat epäotolliset ja puupäädyt ovat karheita ja siistimättömiä. Aiemmat tutkimukset ovat käyttäneet perinteisiä konenäkömenetelmiä, jotka toimivat huonosti vaihtelevan laatuisiin ja muotoisiin puun päätyihin sekä vuosirenkaisiin.

Tässä työssä kehitetään automaattinen syväoppimiseen perustuva tietokoneohjelmisto vuosirenkaiden tunnistamiseen ja vuosikasvun analysointiin. Ohjelmisto esitellään läpikotaisesti ja sitä verrataan perinteisiin konenäkömenetelmiin. Vertailussa käytettiin oikealta tehtaalta otettua dataa ja syväoppimiseen perustuva järjestelmä suoriutui perinteisiä menetelmiä paremmin.

Avainsanat Konenäkö, syväoppiminen, konvoluutioneuroverkot, semanttinen segmentointi, sahateollisuus, vuosirenkaat

Preface

I would like to thank my thesis supervisor Professor Jukka Suomela and advisor Lic.Sc. Jari Mononen for great guidance in this master's thesis process. I would also like to thank Professor Juho Kannala on guidance in the field of computer vision and deep learning, my colleagues Petri Kalske and Matias Lehtinen, as well as the whole team of the sawmill client who were supportive of this project and thesis.

Special thanks to Lauri Nousiainen on support of the thesis process and my father for all the assistance, proofreading and suggestions. Finally, I would like to thank my beloved Tea for love and support.

Otaniemi, 23.12.2019

Jukka Laakko

Contents

Abbreviations

CLAHE	Contrast Limited Adaptive Histogram Equalization
FFT	Fast Fourier Transformation
DFT	Discrete Fourier Transformation
CNN	Convolutional Neural Network
FCN	Fully Convolutional Network
HoG	Histogram of Oriented Gradients
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
WCE	Weighted Cross-Entropy
GDL	Generalized Dice Loss
IoU	Intersection over Unit

1 Introduction

Analysis of wood growth is a necessary quality control step in a sawmill for determining the structure and load bearing capabilities of the wood. Too fast grown wood cannot be used in all applications and has to be detected and removed from the conveyor before further processing in order to avoid unnecessary wastage. Growing speed is determined by calculating the distances between the annual rings of the wood end-face. Having the annual growth analysis done manually by a human operator is a laborious job and prone to errors, as the planks are moving fast in a process line and spotting millimeter distances is practically impossible. Automatic analysis of annual growth in a sawmill environment would therefore increase the throughput of the factory while reducing wastage. Additionally, removing a worker from the wood growth analysis spot reduces monotonic manual labor at the factory and potentially improves the working conditions of the sawmill.

Automatic analysis of wood growth has been studied before [1, 6, 12, 20, 21, 22], with most of the studies focusing on dendrochronology, the science of studying annual rings for the purposes of research fields such as geology and environmental studies. This is, however, not applicable to the sawmill environment, where the wood end-faces are cut into planks, unclean and in rough condition with various defects and anomalies, such as saw cuts, tar and knots (see Figure 1). Imaging quality in a sawmill suffers from time constraints, vibrating factory lines and varying lighting conditions. By contrast, in a dendrochronology setting, the environment is clinical and the woodblocks remain static, an optimal condition for imaging. Moreover, past studies on annual ring detection at both the sawmill and the dendrochronology setting have used traditional image processing methods which rely on handcrafted features and do not generalize well for different types and shapes of wood. Annual rings have an arbitrary and non-trivial loosely circular shape, thus making the use of image processing techniques difficult, even more challenging with the rough condition wood faces of a sawmill.

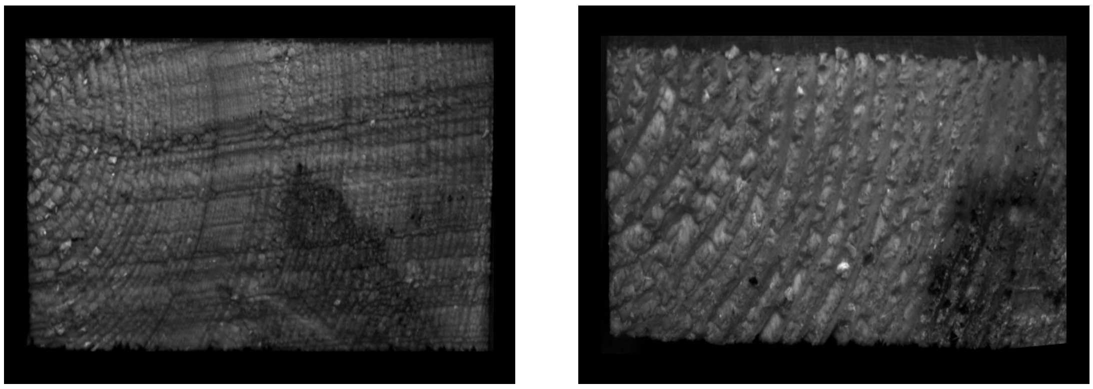


Figure 1: Images of cut and untreated wood end-faces.

One solution to these problems is to use deep learning. In the recent years, deep learning and particularly convolutional neural networks (CNN) have become increasingly popular, revolutionizing the field of machine vision and surpassing traditional image processing methods in many applications similar to annual ring detection such as medical image segmentation [27, 36], and retinal blood vessel segmentation [10, 25]. CNNs [5] simulate the behaviour of human vision through artificial neurons, which are arranged into a network in a fashion similar to that of the visual cortex of the human brain. These artificial neural networks are capable of learning non-trivial features given a ground truth dataset of training and validation images. Developing an image processing based system for detecting annual rings is difficult, whereas annual ring detection by human vision is effortless, which suggests that the use of a deep learning based method would be ideal for this task. Until now, only one study [12] has attempted to use deep learning based methods for annual ring detection, however, it is developed for dendrochronology applications and focused on planks imaged from the top instead of the end-faces, nor did it attempt to count annual rings.

This thesis develops a proof-of-concept implementation of a complete end-to-end annual growth analysis software for a sawmill using deep learning based methods and real data captured from the factory line of that sawmill in order to automate the currently manual task of wood growth analysis and present a general solution for robustly detecting annual rings from all types wood in a sawmill setting. This increases the efficiency of the sawmill and releases a worker for more meaningful tasks. Furthermore, the implementation of this thesis showcases the advantages of using a modern deep learning approach on a problem that has previously not been completely solved in a general case.

The remainder of this thesis is organized as follows. Chapter 2 recaps previous studies on annual ring detection, presents the sawmill and data of this thesis as well as summarizes the architecture of the system and research methods used. Remaining chapters discuss the steps of the solution presented at Chapter 2 in detail. Chapter 3 discusses the annual ring detection using the chosen deep learning method and compares it to traditional image processing methods, Chapter 4 presents an algorithm for finding the pith of the wood, and Chapter 5 discusses the calculation of annual ring distances upon finding the annual rings in Chapter 3 and the pith in Chapter 4. Finally, conclusions of this thesis from both practical and academic perspective as well as discussion on future work are presented in Chapter 6.

2 Overview

This chapter presents an overview of the problem of this thesis. The problem statement was introduced in the previous introduction chapter, along with motivation for the thesis. This chapter is organized as follows: Section 2.1 explores and reviews previous studies that are relevant in terms of this thesis, Section 2.2 describes the setting and hardware of the sawmill in more detail, Section 2.3 presents a graph of the proposed system and its requirements, and Section 2.4 looks into the raw image data and its preprocessing.

2.1 Background

Computer vision applications in the sawmill industry [20] are related to quality control as the quality of the wood denotes its strength and load-bearing capabilities. Too fast grown wood can however be used in some applications, therefore it is vital to be able to differentiate between fast and normal grown wood early in the sawmill factory line in order to make use of all the wood material and reduce wastage.

There are not many studies on annual ring detection in the sawmill environment but the topic has been explored more in terms of dendrochronology, the science of studying tree rings for the purposes of scientific fields such as geology, climatology and environmental studies [12]. The main differences between dendrochronology setting compared to that of sawmill are reviewed in Table 1 based on the papers reviewed in this section as well as on the author's observations.

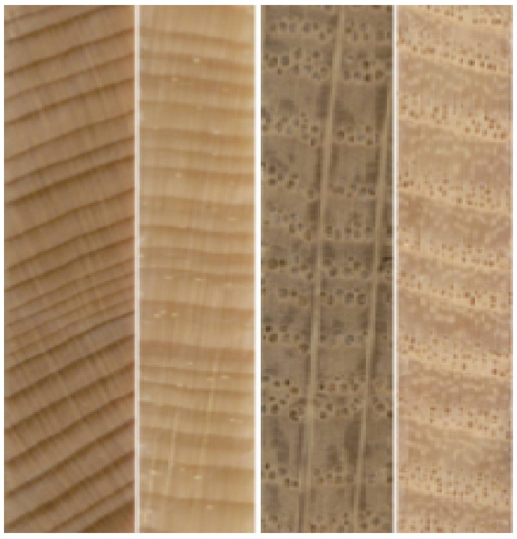
Annual ring detection. To the best of the author's knowledge, the only paper besides this thesis that does annual ring detection using deep learning based methods is Fabijanksa and Danek's DeepDendro [12], which addresses the problems with previous methods that have used traditional image processing which rely on handcrafted features, suffer from limited accuracy and strict restriction on the shape and type of the annual rings. DeepDendro provides a generic and automatic solution for tree ring detection for dendrochronology purposes. Their system uses convolutional neural network, a customized version of U-Net [27], trained on a dataset of 75 wood core images labeled by a dendrochronology expert and were further split into small patches of 80 000 images of which 80 percent were used for training. U-Net is a popular convolutional neural network architecture that makes use of skip connections which forms an U-shaped organization of layers and improves the recovery of fine-grained detail that is important in segmentation of thin lines. Detailed explanation of U-Net and other popular neural network models are presented in Section 3.2.2. The patches were chosen randomly with a specific algorithm, which ensured that all patches contain tree rings since the background area is dominant in the pictures. The images of wood cores that were taken with a scanner and had an average resolution of 9713×172 pixels. As postprocessing, they thresholded the segmented images by the strongest responses and applied iterative thinning and cleaning on the binary images to obtain thin and clear tree ring boundary lines. They experimented with various training options and parameters to further improve the results. Their proposed

	Dendrochronology	**Sawmill Industry**
Environment	clinical, clean, good lighting	industrial, dusty, varying lighting
Wood	hand-picked, good condition	arbitrary, varying condition, defects & saw cuts
Speed target	faster than manual analysis by scientist, minutes	as fast as the process line, seconds
Data	good quality images	varying quality images (dust, process line vibrating)
Application	science (climatology, geology, etc.)	quality control for wood product manufacturing
Cost of error	false results on a scientific study	wastage, expenses
Automation benefits	speed, reduce manual labor	increase efficiency and quality, reduce wastage, reduce manual labor

Table 1: Comparison between dendrochronology and sawmill setting on automatic annual ring detection and analysis.

system was rather successful and achieved a 96 percent annual ring detection rate. Another major advantage of DeepDendro is that it is a black box system which does not require any parameter tuning thus provides a generic solution on the problem. The results of Fabijanska and Danek's paper serve as motivation for the annual ring detection part of this thesis, although its application differs: it uses images of wood cores instead of roughly sawed wood end-faces and the images were taken on a clinical environment with a scanner whereas the images in this thesis are taken on a generic industrial camera on a sawmill setting. The data of [12] and this thesis are compared in Figure 2.

Norell's doctoral thesis [22] on automatic log end-face analysis contains a comprehensive survey on previous studies on annual ring detection as well as recaps multiple papers related to the subject, some of which are discussed in this thesis as well, in Sections 3.1, 4 and 5.2. It thoroughly covers the whole log end analysis process from image acquisition to preprocessing, theory to applications. Norell's thesis was a definitive background material for the image processing parts of this thesis, although the main focus in this case is in the deep learning based approach. Another comprehensive reference on digital image processing is Gonzales and Woods's [14] book.


(a) Wood cores scanned from top in a dendrochronology setting

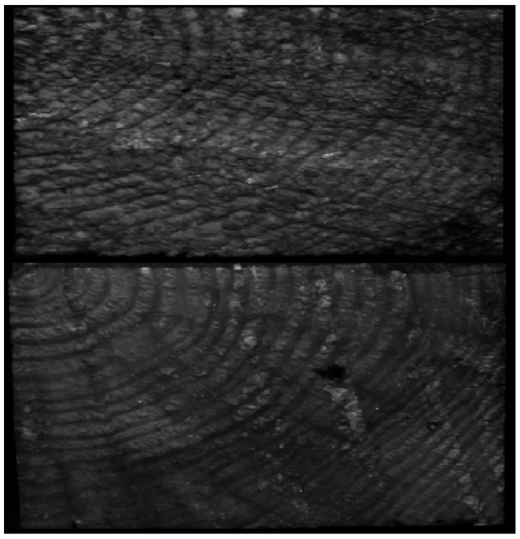

(b) Wood end-faces imaged from the front in a sawmill setting

Figure 2: Comparison between the data of DeepDendro [12] and this thesis.

Related applications. Besides research on annual ring detection, studies on somewhat similar tasks such as blood vessel detection, medical image segmentation, satellite image segmentation and fingerprint enhancement were explored.

Similarly to annual rings, blood vessels in retina images have thin lines and nontrivial elliptical shapes. Localization of retina blood vessels is important in diagnosis of diseases, doing it manually is time consuming and tedious, thus blood vessel segmentation is a widely researched subject [25]. Multiple studies have used deep learning based methods for the blood vessel segmentation task and surpassed traditional image processing methods, which further motivates the usage of convolutional neural networks for the annual ring detection of this thesis. Furthermore, the research on applying neural networks on blood vessel segmentation can provide insight on the neural network architecture for this thesis.

Peng et al. [25] used a custom neural network model based on U-Net for retinal blood vessel segmentation with the highest accuracy on DRIVE, a popular retinal image dataset, as of 2018, surpassing traditional image processing methods. Their network architecture called CDNet is symmetrical in shape to U-Net but proposes to use different kind of segmentation blocks with short propagation paths in order to recover more information from the propagation step. Additionally, they used batch normalization, dropouts, and rectified linear units (ReLU) in their segmentation blocks to improve the training process. These modifications to U-Net were proved to improve accuracy, sensitivity and specificity metrics in their evaluation of segmentation results on the DRIVE dataset. Explanations of neural network layers such as convolution, batch normalization and ReLU layers as well as discussion on deep learning in general is found in Section 3.2.

A similar study by Dasgupta and Singh [10] published a year earlier used a neural network for the retinal segmentation task on the same DRIVE dataset as well with a slightly lower accuracy, further indicating the superiority of deep learning based methods on a non-trivial segmentation task as opposed to the traditional image processing methods. Their implementation is a simple fully convolutional neural network model which consists of two convolutional layers with 32 filters, a max-pooling layer, two convolutional layers with 64 filters, an upsampling layer and two convolutional layers with 32 layers. Rectified Linear Units and dropouts are used after all layers except the last layer uses a softmax layer.

Sedov et al. [28] implemented a neural network for semantic segmentation of buildings in satellite images and explored effects of different loss functions in the training process. While buildings in satellite images do not look like annual rings, the segmentation task shares similar problems as with annual ring detection: the objects are non-trivial in shape and size, there are small details that should be found, and there is not much training data available. They used a neural network derived from U-Net that was modified to have separate encoders for RGB and near-infrared components of the satellite image. U-Net was chosen as the network architecture as it had performed the best in their previous research of satellite image segmentation in comparison to other convolutional neural network models they tried. The network was trained on Inria dataset, which contains 180 satellite images that covers 810 square kilometers of ground in total. The dataset is pixel-wise labeled so that each pixel belongs to either ‘building’ or ‘not building’. They compared four different loss functions in their study which could provide valuable information in the loss function choice for the semantic segmentation task of this thesis.

Fingerprint recognition and enhancement is a comprehensively studied subject in the field of image processing due to its importance in police work and commercial applications such as biometric security methods [34]. Fingerprint images look similar to binarized images of wood faces, with dark circular lines on a bright background, thus similar methods could be used as post-processing in enhancing and reconstructing broken annual rings in the binary images of wood faces after the annual ring detection. Fingerprint detection is also dealing with a related problem of faint, noisy and broken lines, especially on the fingerprints of elderly people and manual workers [34]. Studies of enhancing low-quality fingerprint images and reconstructing broken ridge lines have used methods such as directional Gabor filtering [16], Fourier domain filtering [35] and FFT-based methods [34] to successfully enhance faint and broken lines.

Pith detection. In the literature, pith detection has often been studied alongside annual ring detection, as it is required in order to calculate the annual ring distances, which is usually the end-goal in sawmill and dendrochronology applications alike. Pith is the center, or core, of the wood end-face and its annual ring pattern. As stated earlier, images in dendrochronology studies are in good condition with clear circular ring pattern and speed requirement for pith detection algorithms in dendrochronology is not strict. Therefore, pith localization techniques used in dendrochronology would not be applicable on a sawmill setting with arbitrary shaped annual rings, wood faces of rough condition and the requirement for fast processing. According to a literature review in [22], the approach for pith detection is similar in most studies, they use information about the annual ring orientation in order to find the center point by finding out where the directions are pointing.

Andreu and Rinnhofer enhanced annual rings and presented an algorithm for pith localizing on X-ray based computer tomography images in the sawmill industry [1]. Computer tomography scanners are widely used in medical applications but according to Andreu and Rinnhofer such scanners are not used in sawmills, at least at the time of their study in 2002. The industrial scanner they used was capable of taking one wood image per second, thus too slow for volume-oriented sawmills but sufficient for quality-oriented sawmills. After enhancing the annual ring pattern by acquiring orientation information in the Fourier domain and then using directional Gabor filtering, the images had fairly clear annual ring pattern. To find the pith locations, they used a Hough Transformation based method. Generalized Hough Transformation is somewhat robust to noise and shapes but suffers from computational inefficiency when used with high sensitivity settings. To overcome this issue, search space should be reduced or decomposed. They used the fact that a line which is bisecting any chord of the circle passes through the center point. Each of these chords were transformed into lines in parameter space and all points in the lines are candidates for being the at the center point and the maximum value at the parameter space is the position of the pith. This method proved to find the center point rather robustly.

Norell and Borgefors's paper from 2008 [23] claims to be the first at estimating the position of the pith from images of rough and unpolished wood end-faces in the sawmill environment using a generic digital camera. The images were taken on manual settings in a resolution of 2046×1534 pixels, depicting different characteristics of sawmill wood. Their solution is based on the fact that annual rings can be approximated as simple signals and their directions are used to build an intersection matrix. For local orientation calculation they implemented two distinct methods, one using Quadrature filter method and other using Laplacian pyramid method. The image is first divided into blocks, local orientations for each pixel are calculated using either of the previously mentioned methods and the orientation with the highest certainty c is chosen to represent each block, unless the certainty is below a threshold value T_c, in which case the block is ignored. Certainty of the block's orientations is calculated as follows:

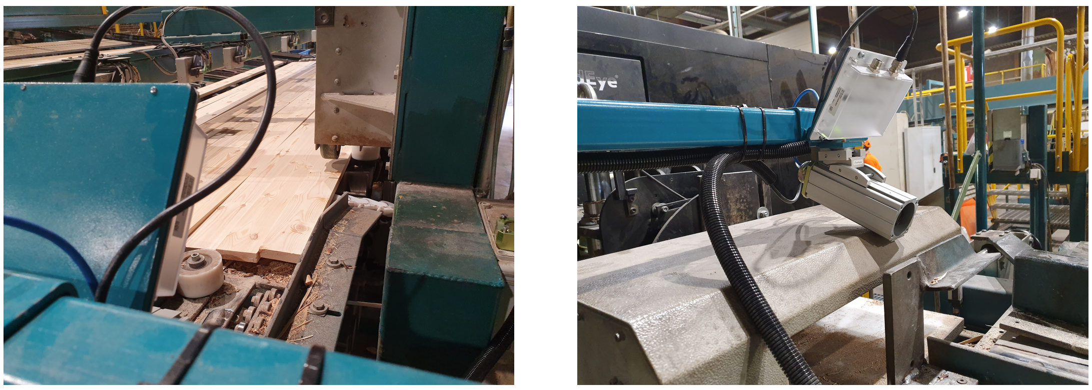
$$c = \frac{\lambda_1 - \lambda_2}{\lambda_2}, \tag{1}$$

where c is the certainty value, λ_1 and λ_2 are eigenvalues acquired from orientation calculations. Block orientations are then used to build the intersection matrix by drawing line from the center of each block towards the orientation. Region around the global maximum of the intersection image is where the pith estimation is located and average of that region is the final pith estimate. The results varied slightly between the Laplacian pyramid method and the Quadrature filter method but both methods were promising and capable of handling digital images of rough and uncleaned wood end-faces. The Laplacian pyramid method performed faster at approximately 2 to 3 seconds on an unspecified standard PC, making it a more viable option for a real-time application in a sawmill. It should be reminded that their study was done on 2008, the running time should be considerably faster on modern hardware.

2.2 Sawmill Setting

The sawmill of this thesis is in the process of automating tasks that, at the moment, require human operation, counting of annual ring distances, e.g., determining wood's yearly growth, being one of these tasks. As stated in the introduction, too fast grown wood cannot be used in all products due to its weaker structure; therefore it is of importance to send these fast grown wood onto a different product line where they can be used. Failure to spot these pieces of wood and have them remain in the line which processes normal grown wood causes unnecessary wastage, as the fast grown wood has to be thrown out if it is processed further in that product line.

The place in process line where the automatic year growth analysis takes place is before cutting and cleaning of the log end-faces. The woodblocks move fairly fast, approximately one plank per second passes the observation spot. There are various sizes of blocks that are processed in this stage of the process line, most commonly 150 × 100 mm and 50 × 50 mm. The camera is an industrial network camera manufactured by Basler, with a maximum resolution of 1920 × 1200 pixels. There is no zoom functionality, and the woodblock occupies roughly half of the resulting image, which is a sufficient resolution for this project. The camera is tightly secured in a spot above the factory line, as seen in Figure 3. There are bright lights above the wood logs in the spot where the imaging takes place and as a results of that, the wood end-faces are shaded, which is not desired. The shaded end-faces can be seen in Figure 3. IR filters cut off other visible rays of light except for the red-light infrared spectrum, which makes nearby objects bright and background dark, in this case makes annual rings in a wood end-face stand out more. Therefore, an IR filter pointing at the wood faces was installed above the camera. Using an IR filter also makes the image appear grayscale, which does not matter in this case as color information is not meaningful in detecting annual rings. There is also a laser-based hardware trigger; when the wood passes the trigger, the camera takes an image. While the trigger is efficient, the photos are not still taken exactly at the same spot every time, which is likely due to limitations in the speed of camera hardware or software.


(a) Behind view of the camera

(b) Front view of the camera

Figure 3: Camera setup in the year growth analysis spot.

The camera is connected to a nearby computer in the factory running Ubuntu 16.04 LTS. There is an accompanying camera software installed on the computer, which is used for controlling camera settings, including brightness, gamma, contrast and exposure time. Camera manufacturer's API will later be used to run the annual growth calculation system proposed in this thesis in real-time. Later there will also be a user interface, which the factory workers can use to monitor the annual growth verifications done by the system, but that is not in the scope of this thesis.

Verifying annual growth from the wood end-faces by human eye is prone to errors, as the blocks move fast in the factory line and it is impossible to spot a precise threshold (more than 6 millimeters between two rings counts as too fast grown) by the human eye. Therefore, classifying wood blocks as sufficient or too fast grown is at the moment done only approximately. According to the factory workers, this task is also quite tedious and tiresome work, and they would rather be doing some other tasks. Therefore, automatic verification of year growth would drastically increase the accuracy of the process and thus reduce wastage and increase profits as well as potentially increase working conditions of the sawmill.

2.3 Approach

The proposed software for automatic analysis of wood annual growth consists of several distinct steps, as depicted on the block diagram of Figure 4. These steps are presented in detail throughout this thesis: Section 2.4 discusses input data and its preprocessing, Section 3.2 describes annual ring detection using convolutional neural networks, Chapter 4 presents an algorithm for pith detection and finally, Chapter 5 discusses the calculation of annual ring distances. Appendix B includes example runs of the whole system. Requirements of the system are:

- Sufficiently accurate
- Can handle oddly shaped rings, saw marks, noise, cracks, different lighting conditions, etc.
- Applicable to blocks of different sizes
- Can adapt to changes in camera position and imaging settings
- The total running time should be no more than 1 second
- Design should be modular and easily modifiable, fixable and expandable.

The system implemented in this thesis is a proof-of-concept level implementation, and therefore does not reflect the final commercial production system, however, the software of this thesis uses real sawmill data and should indicate that the system requirements are feasible.

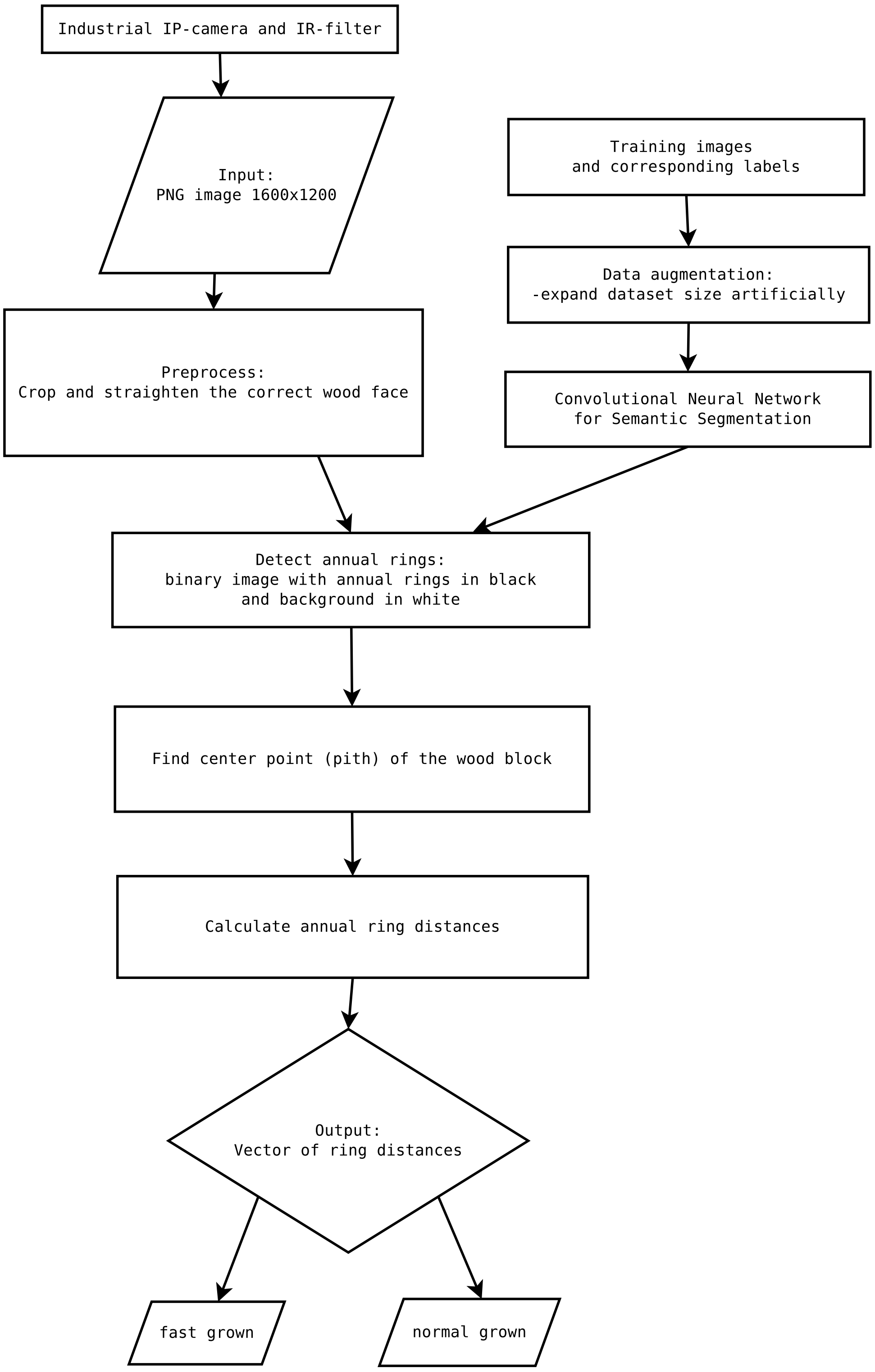

Industrial IP-camera and IR-filter
Input:
PNG image 1600x1200
Training images
and corresponding labels
Data augmentation:
-expand dataset size artificially
Preprocess:
Crop and straighten the correct wood face
Convolutional Neural Network
for Semantic Segmentation
Detect annual rings:
binary image with annual rings in black
and background in white
Find center point (pith) of the wood block
Calculate annual ring distances
Output:
Vector of ring distances
fast grown
normal grown

Figure 4: Block diagram of the system architecture.

2.4 Data

The data of this thesis is continuous flow of pictures of wood log faces captured from the factory line in a png format. Figure 5 shows an unprocessed image straight from the factory line. The sawmill's camera setup discussed in Section 2.2 produces good quality images for this task after fiddling with camera settings in the proprietary software. The images appear sharp with decent contrast and exposure. The annual rings are as clearly visible as they can be, given the rough condition of the wood faces.

After obtaining the image from the factory line, it is converted from RGB color space to grayscale. As the images already appear grayscale, color does not provide any extra information to exploit in this case. Digital images in RGB format are matrices of three layers, where the three layers represent values of red, green and blue channels, whereas a digital image in grayscale color space is a single matrix of intensity values ranging from 0 to 255. This means that grayscale images are more simple and easier to understand and visualize. Computation is also faster on greyscale images as there are three times less values to process.

The data is then preprocessed so that the rightmost woodblock is extracted from the image and copied on a blank canvas of 1000×500 pixels. The rightmost block is always noticeably brighter than any other block present in the image due to the IR filter pointing at it. Therefore, thresholding can be used to to segment and extract the correct wood end-face from the background. Thresholding [14] is a method of segmenting a grayscale image into binary colors by replacing each pixel in the image by a white pixel if the pixel's intensity value is above a fixed threshold constant T, and to black if it is less than T:

$$g(x, y) = \begin{cases} 1, & \text{if } f(x, y) > T, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

If the image has a histogram with a clear and sharp difference between the peaks representing the foreground and the background, then the threshold value can be easily chosen manually or programmatically by analyzing the histogram. However, this is often not the case in real life pictures, where the difference is not that sharp and there is noise. Additionally, when processing multiple images, the intensities of the images vary unless taken in an exact same position with the same lighting and camera settings. Choosing a fixed value is not an option in this case either as the lighting conditions and the brightness of the wood end-faces vary. The photos are also not taken at the exact same spot every time, resulting in varying intensities based on how close the block is to the IR filter.

Otsu's method [24] was used to choose the threshold value automatically. It is a simple, general, unsupervised, and nonparametric method for selecting an optimal threshold value based on global properties of the histogram instead of looking at local neighboring pixels, maximal gray-level differences, or derivatives. The optimal threshold value is determined by maximizing the measure of separability of classes in gray levels by the discriminant criterion, utilizing zeroth and first-order cumulative moments.

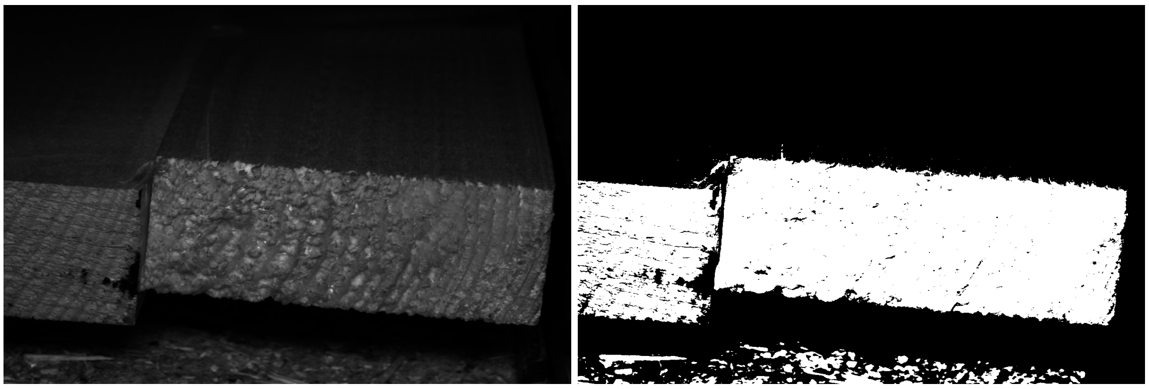

(a) Input image (b) Binarized with Otsu's method

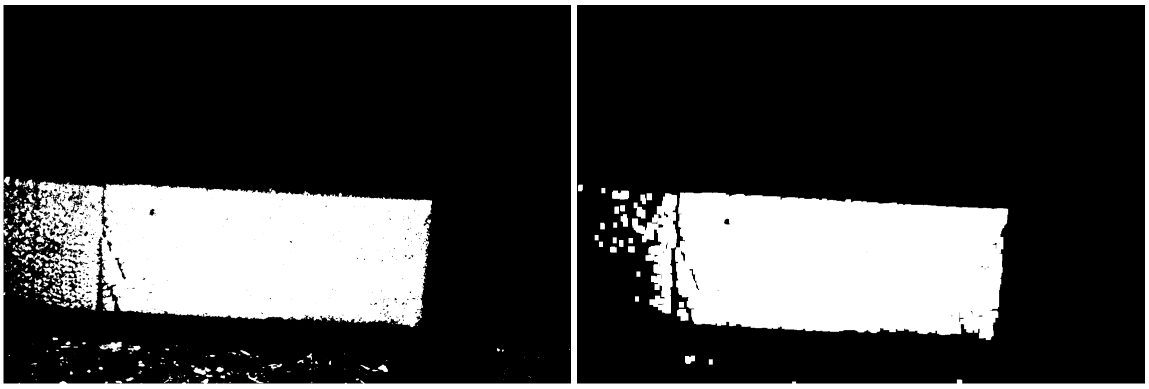
Figure 5: Image before and after thresholding.

Sometimes the woodblocks were slightly touching each other, which resulted in the thresholding to pick too large an object. This was fixed by applying morphological opening on the image, which was sufficient to separate most of the falsely combined objects. Mathematical morphology [14] is a set theory approach for digital image processing, where binary images are viewed as sets in Z^2 integer space, and each element is a tuple with (x, y) coordinates of either white or black pixel in the binary image. Morphological operations can be used to manipulate white pixel (or black pixel, depending on choice) areas of binary images. Fundamental morphological operations are erosion and dilation, which are used for shrinking and expanding, respectively. Erosion of A by B is the set of points z such that B is contained in A once translated by z, where A is the set of white pixels, B is a structuring element. Dilation of A by B is set of reflecting B above origin by z, so that B and A overlap. Opening operation is erosion followed by a dilation, defined as [14]:

$$A \circ B = (A \ominus B) \oplus B, \tag{3}$$

where $\ominus$ is erosion and $\oplus$ is dilation. Structuring element denotes the shape of the operation to apply on the set, in this case a rectangular block with a height of three pixels and a width of two pixels. Opening is useful for removing noise and small objects, and in this case separating objects where a small number of white pixels in two woodblock objects were touching each other.

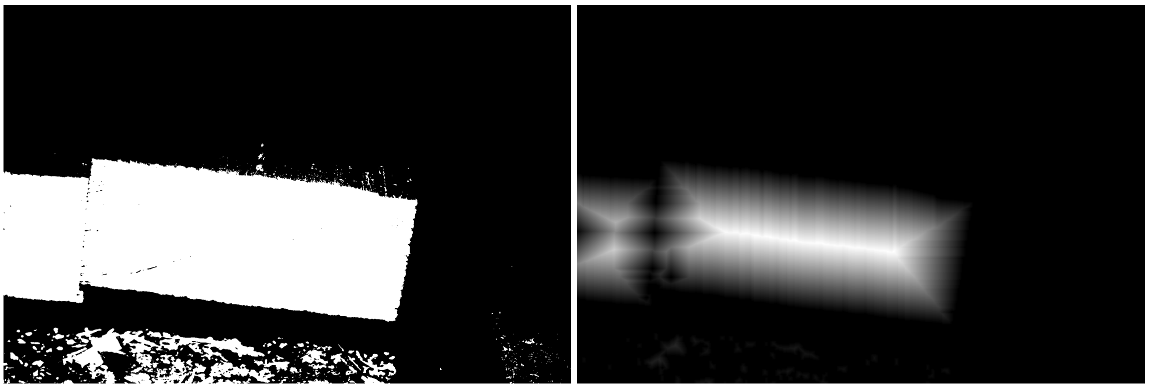
While morphological opening was sufficient for the cases where two wood block objects were slightly touching each other, it cannot be used in a case where the objects are completely overlapping, as in Figure 7. This case was instead handled with distance transform [4] followed by another Otsu's thresholding. Distance transform calculates each pixel's distance to the closest zero pixel. It finds the shortest path to the zero pixel by performing shifts (horizontal, vertical, diagonal or knight's move) over a local neighborhood. A three by three neighborhood was chosen due to its speed over a larger mask and the formula used for calculating distances is $|x_1 - x_2| + |y_1 - y_2|$. Strong distances appear brighter in the resulting distance transformed image, whereas the part where blocks are overlapping is dimmer.



(a) Binarized with Otsu's method (b) Morphological opening applied

Figure 6: Opening separates slightly touching objects.

Objects could thus be separated by applying Otsu's thresholding on the distance transformed image. Additional morphological dilation was performed to enlarge the objects as they were left slightly smaller after the distance transformation and thresholding. Visualization of the distance transformation on a binary image is shown in Figure 7.



(a) Binarized with Otsu's method (b) Distance transformed image

Figure 7: Distance transform separates completely overlapping objects.

After thresholding and separation of overlapping objects, contours were searched in the resulting binary image, and the list of contours was filtered based on their area. Knowing that the wood block object has to be larger than 2500 mm^2, approximately 270 000 pixels in the image, a threshold value such as 200 000 pixels safely ignores small objects and leaves the wood end-faces in the contour list. In case there were multiple sufficiently large objects in the contours list, the rightmost one was chosen based on the x–coordinates of the contours in the list. The woodblocks do not lie completely straight in the factory line. Therefore a rotated rectangle was fitted on the contour, which contains the correct woodblock and straightened based on the angle of this rotated rectangle. Then it is straight forward to crop the straightened rectangle from the image and copy it on a blank, black canvas, as shown on Figure 8.

The canvas size was chosen to be 1000×500 pixels as each of the wood sizes going through this point of the factory line fits in it nicely. Appendix A contains a collage of these extracted wood end-faces, a comprehensive overview of the data of this thesis.

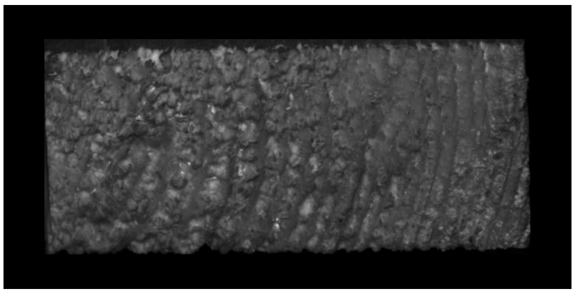


Figure 8: Image after straightening and cropping.

3 Annual Ring Detection

In order to calculate the annual growth of the woodblocks, it is necessary to detect and segment annual rings from the wood end-faces. The shape, size and condition of the annual rings varies. There are also often saw cuts and defects such as knots or tar that can easily interfere with the detection. These facts make the segmentation task difficult. If all the woodblocks would have a similar condition, then it would be possible to handcraft a solution, but this is not the case here. Instead, a solution that generalizes well for various types of wood has to be designed.

Most of the previous studies focused on dendrochronology, where the studied wood faces are in good condition with no defects and with a clear annual ring pattern. Until now, only one study [12] had attempted to use deep learning for annual ring detection, and provided a general solution to the problem. However, as stated in the literature review of Section 2.1, the data of this Fabijanska and Danek's approach differs from that of this thesis as it uses clean and hand-picked images of wood cores instead of arbitrary, rough, and unclean images of wood end-faces. Moreover, the images were taken in a clinical environment on a scanner whereas the data of this thesis was taken on a generic industrial camera on a sawmill environment. Despite the differences, the underlying idea of using CNNs for the detection task is similar.

As stated earlier, the condition of wood faces is rough and uncleaned, there are defects, dirt and saw cuts, and the annual rings are complex and characteristic with a lot of variation. Additionally, the wood faces in this thesis are cut to pieces so a full circular annual ring pattern is not visible. Therefore, it is a complicated task to create a general algorithm for reliably detecting annual rings for the data of this thesis by hand, however CNNs are good at learning and generalizing even non-trivial features. For the sake of comparison, this chapter implements both an image processing and a deep learning solution to the problem of which the latter is used in the final software of this thesis. Deep learning has revolutionized computer vision and in some cases, in fact, it even surpass human vision. This becomes apparent when comparing the results of the traditional image processing methods of Section 3.1 with the results of deep learning based method of Section 3.2.

3.1 Image Processing Methods

Tree ring detection is an old area of research in the field of machine vision, and a reliable solution has been searched for over a decade [20]. These studies use digital image processing methods such as edge and boundary detection, which rely heavily on input data being uniform as they must be partly handcrafted, thus tree rings of unusual shapes can not be properly calculated [12]. These methods also suffer from noise, and missing gaps from tree rings need to be reconstructed.

Section 3.1.1 describes the preprocessing of the data, Section 3.1.2 discusses edge detection, Section 3.1.3 presents postprocessing methods, and finally, Section 3.1.4 discusses the results of using image processing methods to detect the annual rings on the data of this thesis.

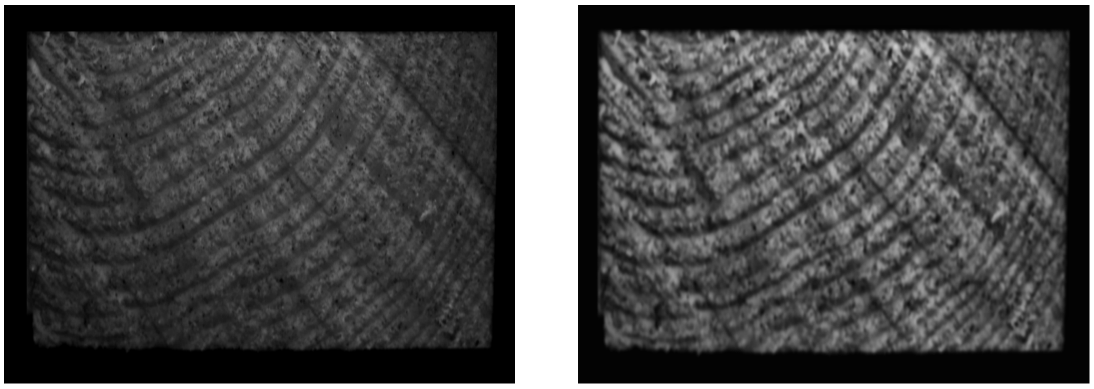
3.1.1 Preprocessing

The first step of the tree ring segmentation process is to preprocess the images to attenuate noise and unwanted features and to highlight the desired features, that is the annual rings. Having noise reduced makes it easier to perform edge detection later. Noise can be removed by applying a smoothing filter on the image, however sometimes smoothing may also blur edges too much, which makes it more challenging to detect annual rings. Bilateral filter [33] is a type of smoothing filter that preserves edges while blurring the background, which is needed for this task. It is a non-iterative and a simpler method than similar edge preserving blurring techniques such as Anisotropic diffusion. While it is simpler than other edge-preserving methods, it is still more complex and substantially slower than simple kernel filter based smoothing techniques such as Gaussian blurring. However, the overall running time of preprocessing step was still kept within reasonable limits; thus it was safe to use in this case. Bilateral filtering works by combining range and domain filtering. Range filters operate in the range of an image and measure the similarity between nearby pixels, whereas traditional filters do in the domain. Combining the two filtering methods means replacing each pixel with an average of nearby and similar pixels, which causes edge pixels to be blurred less than background pixels, therefore, being edge-preserving as desired.

In addition to smoothing, a small amount of contrast enhancement using Contrast Limited Adaptive Histogram Equalization (CLAHE) algorithm [37] was done to make edges stand out more. Problem with standard histogram equalization techniques is that in some cases, they may cause the undesired parts of the images to be highlighted if they appear as significant peaks in the image histogram, thus actually making the output image worse than the original. CLAHE presents an improvement; it aims to preserve and enhance edges while improving overall contrast of the images by using local regions rather than inspecting global histogram as in the regular method. The image is first divided into rectangular grids of desired size in which the contrast is calculated. Histograms are then calculated for each rectangle, and contrast for each of these local rectangular regions is optimized by the cumulative distribution function. Furthermore, contrast is limited by clipping the histogram so that only a certain number of pixels in local histogram bins are allowed. Clipped pixels are then distributed so that the total histogram count remains the same. The purpose of this contrast limiting is to prevent amplified background noise.

While the previously discussed smoothing and contrast enhancement techniques make the annual rings more visible, they also make defects and saw cuts more visible, therefore should be used with moderation. An example of applying these methods on an image is shown in Figure 9.

Before detecting the edges in the preprocessed images, saw cuts should be filtered out as they would otherwise interfere with the annual ring detection. In a 2010 study, Norell filtered saw cuts [21] by making use of the fact that saw cuts follow a periodic pattern that can be filtered in the frequency domain. The orientation of sawings is constant and repeating, which suggests that it would be visible in the corresponding Fourier spectrum as a rather clear line of high energy. The saw marks



(a) Input image (b) Preprocessed image

Figure 9: Image before and after preprocessing with bilateral filtering and CLAHE.

can thus be reduced by suppressing the energy belonging to saw cuts in the Fourier domain and then transforming back to the spatial domain. Suppression is done by pixel-wise multiplication with a line-shaped filter that covers the high-intensity line in the Fourier domain caused by the saw cuts but leaves the important center region intact.

Fourier transformation is continuous, ranging from $-\infty$ to ∞. In reality, there are finite data samples and Discrete Fourier Transformation (DFT) is used to transform a finite sequence of numbers into another finite sequence of numbers corresponding to the DFT of the original samples. Two-dimensional DFT with finite number of equally spaced samples $F(u, v)$ is expressed as [14]

$$F(u, v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(x, y) e^{-j2\pi(ux/M+vy/N)}, \tag{4}$$

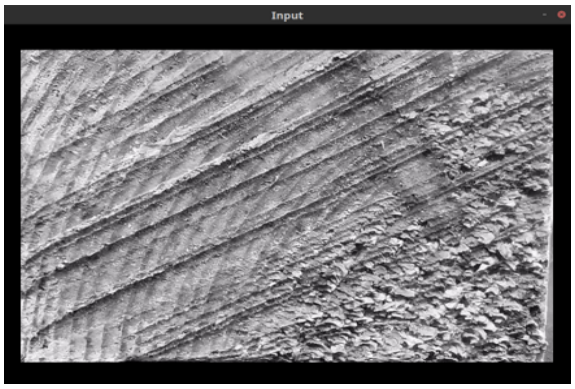
where $f(x, y)$ is an input image of size $M \times N$, u and v are discrete variables in the range of $u = 0, 1, 2, \ldots, M-1$, and $v = 0, 1, 2, \ldots, N-1$.

Correspondingly, inverse DFT for recovering $f(x, y)$ for $x = 0, 1, 2, \ldots, M-1$, and $y = 0, 1, 2, \ldots, N-1$, given the Fourier transformed sample set F is defined as

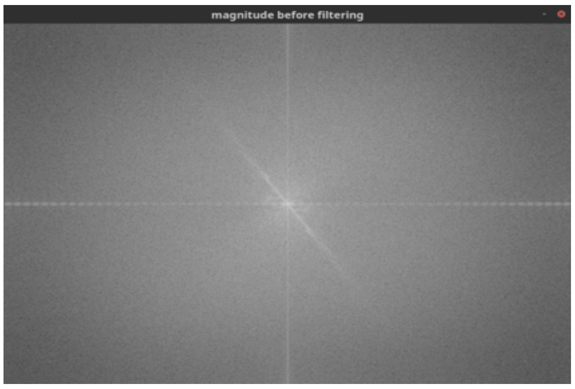
$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} F(u, v) e^{j2\pi(ux/M+vy/N)}. \tag{5}$$

Transformation to Fourier domain in practical applications, such as in this case, is done with a Fast Fourier Transformation (FFT), which is an efficient algorithm for computing the previously defined Discrete Fourier Transformation. FFT essentially reduces the computational complexity from a quadratic running time $O(N^2)$ of brute force DFT implementation (where N is the image size) to a considerably faster logarithmic running time of $O(N \log N)$. FFT is not a specific algorithm, instead there are various different algorithms for obtaining the $O(N \log N)$ running time, that are considered FFT algorithms. Implementation details of some specific FFT algorithm are irrelevant for this thesis, but, essentially, they use various common techniques for designing efficient algorithms such as divide and conquer or factorization.

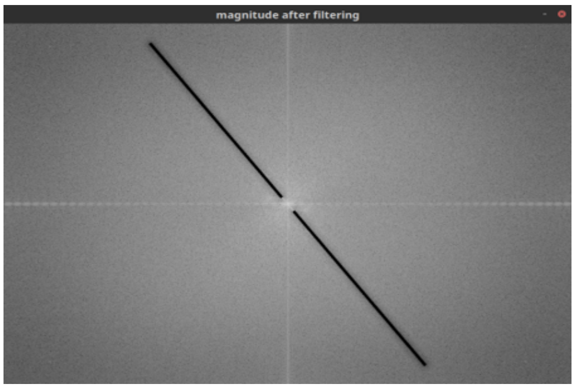
The resulting spectrum after transforming the image with visible saw cuts pattern to the Fourier domain using FFT is shown in Figure 10. The direction of the sawing pattern appears as a line in the opposite direction (of the actual pattern direction) at the Fourier spectrum, and it can be determined by calculating the total energy in different directions from the spectrum center and then by finding the peak [21]. A suitable filter is a line-shaped filter that covers the line in the spectrum caused by saw cut pattern while ignoring the center area of the spectrum. Furthermore, a Gaussian blur filter with a standard deviation of $\sigma = 15$ was applied on the line filter to make the effect more smooth. Such a filter is shown in Figure 10. The filter is then applied by performing a pixel-wise multiplication with the Fourier transformed image. Finally, the image is transformed back to the spatial domain by performing an inverse FFT. An example of using the Fourier filtering method for reducing saw marks in the images of this thesis is shown in Figure 10; the saw cuts are greatly reduced.



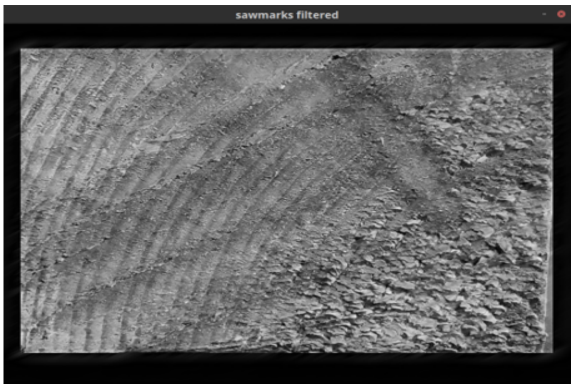
(a) Image before filtering



(b) Fourier spectrum of the image



(c) Line-shaped filter applied



(d) Image after filtering

Figure 10: Visualization of applying a linear filter in the Fourier domain. Saw mark pattern appears as a diagonal line in the opposite direction in the Fourier spectrum.

3.1.2 Edge Detection

The actual annual ring detection can now be done after the previously described preprocessing steps. Extraction of regions of interest in the field of image processing and computer vision is called image segmentation [14], which is arguably the most difficult task when dealing with non-trivial features such as tree rings. It is also the most crucial step, as it determines the success of the remaining annual growth analysis. Most of the common methods for image segmentation are based on the discontinuity and the similarity of neighboring intensity values. In other words, either segment the image based on sharp changes in intensity (edge detection) or do the segmentation based on the similarity of different regions in the image according to some criteria (thresholding). Below is a brief introduction to the mathematical background of some common segmentation methods and discussion of results on applying these techniques on the images of this thesis.

Local changes in intensity are the edges of an image and a line is an edge where there is a sharp change in intensity to either darker or brighter on both sides of the edge. Edge detectors are a common tool for detecting edges and lines, which in this case include the tree rings. Changes in intensity can be found by using first- or second-order derivatives. The direction and magnitude of edges at a pixel (x, y) in an image f can be calculated from its gradient ∇f as follows [14]:

$$\nabla f = \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}, \tag{6}$$

where $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ are partial derivatives at each pixel of the input image and g_x and g_y are directional gradients of the same size as the input image. From the directional gradients calculated by formula x we can derive magnitude M at pixel (x, y), given that gradient direction points at the largest rate of change:

$$M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}. \tag{7}$$

Following the same train of thought, the direction of the gradient at pixel (x, y) with respect to the x–axis is defined as:

$$\alpha(x, y) = \arctan \begin{bmatrix} g_y \\ g_x \end{bmatrix}. \tag{8}$$

The partial derivatives are calculated with kernel masks in order to capture diagonal edge directions. There are several types of masks that can be used, most commonly Roberts, Prewitt and Sobel named after their inventors, each providing slightly different output. Roberts is the oldest method and uses a 2×2 mask while the others use symmetrical 3×3 masks which are better at computing edge directions. The masks act as convolutional matrices, which are slid across an input image in order to obtain the gradients g_x and g_y. The values of these mask matrices are shown in Figure 11.

$$\begin{vmatrix} 0 & -1 \\ 1 & 0 \end{vmatrix}$$

$$\begin{vmatrix} -1 & 0 \\ 0 & 1 \end{vmatrix}$$

(a) Roberts

$$\begin{vmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{vmatrix}$$

$$\begin{vmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{vmatrix}$$

(b) Prewitt

$$\begin{vmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{vmatrix}$$

$$\begin{vmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{vmatrix}$$

(c) Sobel

Figure 11: Common kernel masks for performing edge detection (x–direction on top, y–direction on bottom).

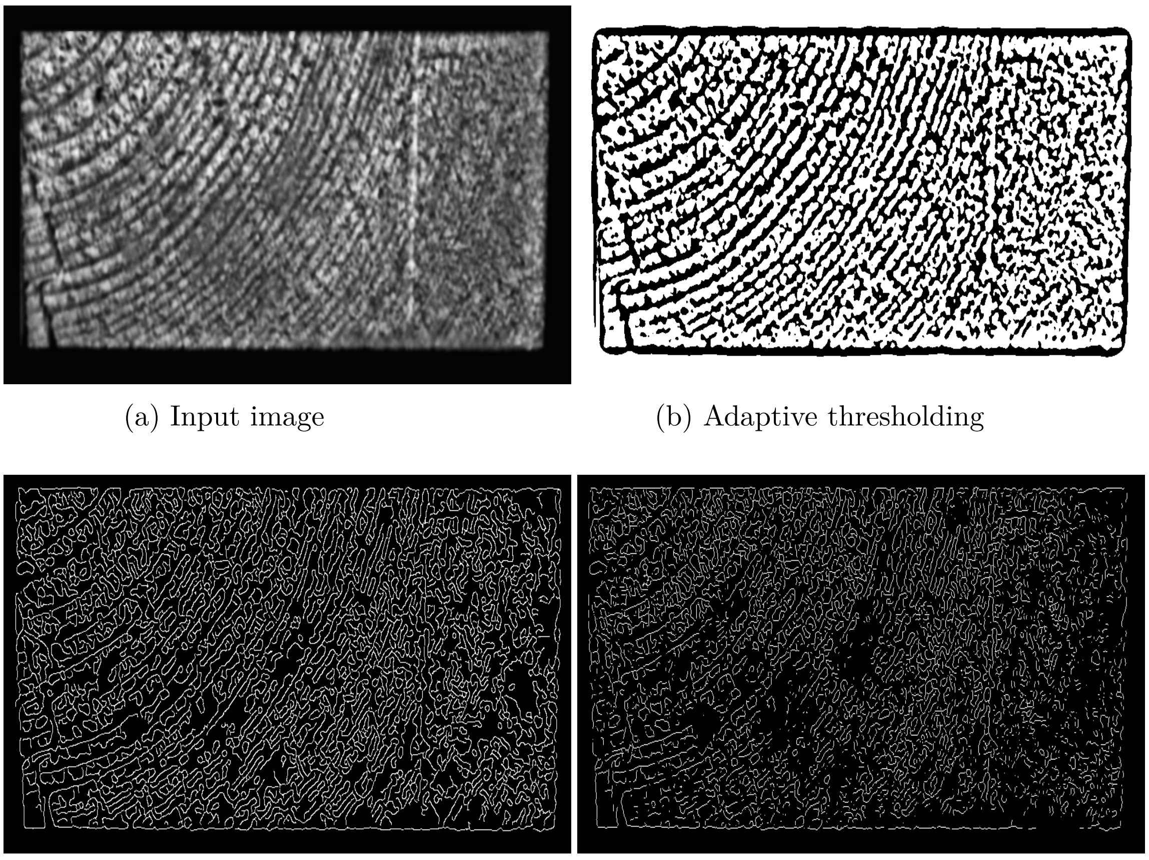
Another common choice for edge detection is the Canny edge detector [14]. It is an extension to the previously discussed edge detectors and therefore slightly more complex but considered more efficient and superior in general. The input image is first smoothed with a circular Gaussian filter, then the gradient direction and magnitude are calculated as in Equation (6–8) with a sliding mask such as Prewitt or Sobel. Lastly, non-maxima suppression or some similar method is applied to make ridges thinner.

Region growing [14] aims to find the desired features directly: it picks seed points and grows a region around them by adding similar neighboring pixels. This is similar to clustering algorithms in data mining and machine learning. Problem with region growing in the case of annual ring detection is the choice of seed points, which is an essential part of this method. If the seed points are wrong, then the resulting segmentation is surely invalid as well. Pixels belonging to annual rings do not have a certain distinct intensity value or shape; thus, it would be difficult, if not impossible, to choose the seed points accurately. It was, therefore, decided not to test this method.

Watershed [14] is another commonly used segmentation method, which is best explained through an analogy of placing a drop of water in a regional minimum spot and having it flood until it hits a dam. The 'depth' which determines the flooding of water is based on the intensity of the image and 'dams' are built to prevent the water from overflowing. The previous analogy is also where watershed gets its name. It is a simple and efficient method for segmentation tasks that contain blob-like objects in a clear background, for example segmentation of coins. However, it was safe to say even without testing that it would not be applicable in this task due to having thin and partly broken annual rings on a rough and uneven background.

Adaptive or variable thresholding [14] is a thresholding method based on local regions of an image. Whereas the basic and Otsu's version of thresholding presented in Section 2.4 is commonly used for separating an object from background, adaptive thresholding does the thresholding in local user-defined regions of an input image thus it can be used for edge detection when using a small window size. The threshold in each local region can be calculated either by a mean of the region or by a Gaussian-weighted sum. A positive constant is then subtracted from the mean or weighted-sum to obtain final thresholding for the region.

Upon testing various edge detection methods with different parameters, it was quickly determined to be an inapplicable technique for this task. The wood end-faces are too rough and therefore contain plenty of edges that do not belong to annual rings resulting in poor results, as seen in Figure 12. Additionally, edges of defects such as a knot or tar would be falsely detected. Fixing this problem would require designing an additional algorithm for finding defects, which would be a difficult (if not impossible) task given the nontrivial shape and size of different defects. Watershed and region growing methods did not work due to previously discussed issues. Local adaptive thresholding proved to be substantially better at detecting the annual rings, although similar problems as those previously described exist. The quality of the segmentation remains below an acceptable level, especially on wood faces with defects or unusually rough conditions. This is the essence of difficulty in image segmentation: it partitions the image into different coherent parts but does not understand these parts. Deep learning based semantic segmentation, on the other hand, tries to learn features of an image through pixel labeled ground truth, therefore being capable of differentiating between an annual ring and a knot, for example.


(a) Input image

(b) Adaptive thresholding

(c) Edge Detection (Canny)

(d) Edge Detection (Sobel)

Figure 12: Examples of different segmentation methods on an input image with CLAHE and bilateral filtering applied.

3.1.3 Postprocessing

A simple approach for postprocessing is the usage of morphological operations, explained in Section 2.4. Of these methods, morphological closing [14], that is dilation followed by erosion, is especially efficient in removing small bits of noise from a white background with a black foreground. User-defined parameters of size, shape, and the number of iterations should be carefully determined, as applying too aggressive closing also causes relevant information to disappear. Applying closing with a circular structuring element of two pixels radius was found to enhance the binary images of annual rings slightly.

As stated in Section 2.1, fingerprint enhancement is a widely studied subject and the shape of fingerprint ridge lines happens to look similar to annual rings. Therefore, similar methods could be applicable in this post-processing step of the binarized wood end-face images. Of these fingerprint enhancement methods, Willis' and Myers' method [34] for enhancing and fixing broken lines in poor quality fingerprint images is particularly elegant in its simplicity. In comparison to enhancement methods based on Gabor filtering, it is considerably faster and less complex. It is based on the fact that directional information of different parts of an image can be found and enforced in the magnitude of Fourier transformation. The image is first divided into blocks of a desired size. A block size of 32 by 32 was used in the original paper to capture three to four ridges and to enable usage of radix-2 FFT. Furthermore, using overlapping blocks was found to improve the results. Then, lines in each block were enhanced by simply multiplying the original FFT by the magnitude of the FFT and then taking inverse FFT of the result to transform back to the original domain. The Willis and Myers FFT-enhancement method is formally defined as follows:

$$g(x, y) = F^{-1}(F(u, v) \cdot |F(u, v)|^k), \tag{9}$$

where F is the Fourier transformation, F^{-1} is the inverse Fourier transform, $g(x, y)$ is the enhanced output and k is a user-defined constant between 1 and 2 for multiplying the magnitude. The larger the value of k the more aggressive the enhancement effect becomes. FFT and the corresponding inverse FFT were defined earlier in this section at Equation (4-5).

Using the method on the images of this thesis produced a similar enhancement effect on blocks with clear ridges. It was capable of fixing some of the broken tree rings and removing noise in good quality blocks but more often than not the fundamental magnitude that was enforced was noise instead of a line belonging to an annual ring. This results in a trade-off between improving clear parts of the image and degrading unclear parts. Figure 13 highlights this effect; it shows the result of FFT enhancement on a binary wood face image where the left part of the image was improved, but the noisy right part of the image had gotten inferior.

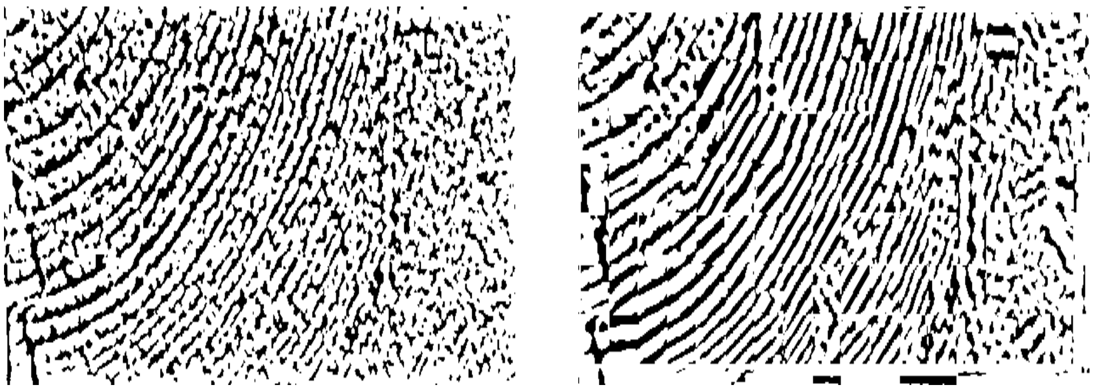

(a) Image before enhancement (b) Image after enhancement

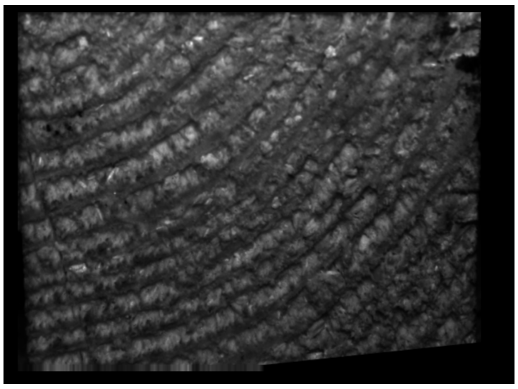
Figure 13: Image before and after FFT enhancement with $k = 1.4$ and block size 32×32.

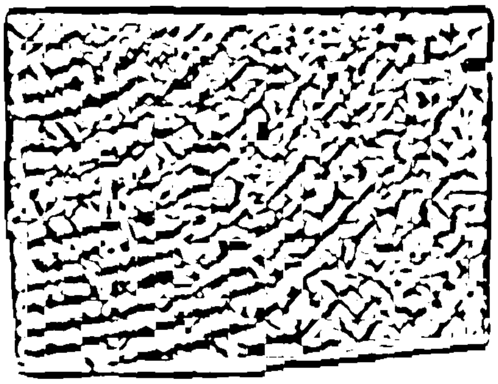
3.1.4 Results

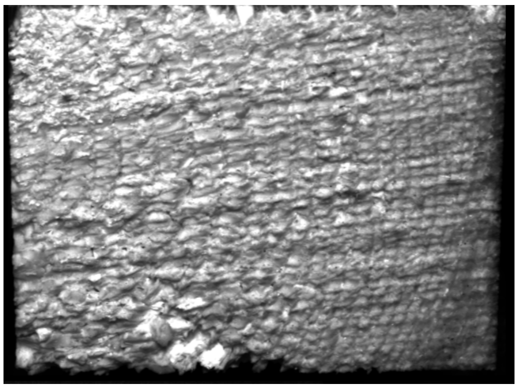
Image processing methods described and implemented in this Section 3.1 were not enough to get a sufficient segmentation except for a small fraction of wood faces that were in good condition with annual rings clearly visible. Additionally, the implementation relies heavily on the images of wood faces that have similar properties as most of the steps presented are hand-crafted. For example, had the size or brightness of the wood faces change drastically, the parameters would have to be tuned again.

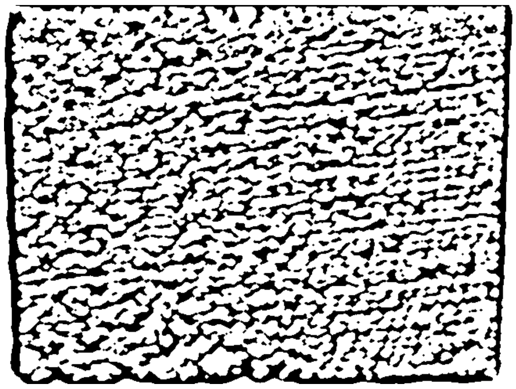
Examples of the annual ring detection implemented in Section 3.1 are shown in Figure 14. The first and third pairs of images show a fairly good input quality image, which has some parts of the annual rings detected but still suffers from noise and a considerable amount of false pixels. The image on the second row is of lesser quality, and the resulting binary image after annual ring detection has practically no usable information in it.

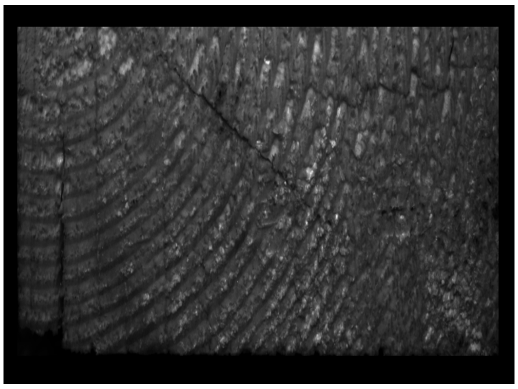
It is difficult to manually develop an algorithm for differentiating between desired and undesired features. While the algorithms could be fiddled with and slightly improved, it would not be worth the time and effort as this type of non-trivial feature extraction task is more suited for a neural network based implementation, which simulates that of human vision. Deep learning based computer vision was not widely used at the time of previous studies on annual ring detection due to a lack of computational power and research. These previous studies were reviewed in Section 2.1. The following Section 3.2 discusses solving the annual ring detection problem with a deep learning approach and shows a truly substantial improvement in accuracy and ease of implementation for this type of detection task compared to the traditional methods.


(a) Input image


(b) Annual rings detected


(c) Input image


(d) Annual rings detected


(e) Input image

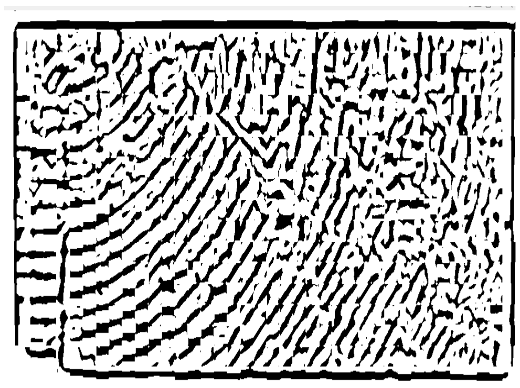

(f) Annual rings detected

Figure 14: Example annual ring detections using image processing methods of Section 3.1.

3.2 Deep Learning Solution

Deep learning [5, 15] is a subset of machine learning where a computer learns concepts through experience by using deep neural networks that are hierarchically organized, so that complex concepts are learned by building on simpler features, rather than specifically instructing the computer of all the needed knowledge. In the recent years, deep learning become widely used and researched in the field of computer vision, which has traditionally been challenging for computers. Vision is effortless for humans; within seconds we can detect objects and their features. This is all handled in the brain through neurons that process the light information captured by eyes. Therefore, it does make sense to try and simulate similar behavior through artificial neural networks. Indeed, deep learning has surpassed traditional image processing methods and in some cases even surpassed human vision. Shortcoming of these traditional methods became apparent in the previous chapter: while the annual rings can be easily detected by human eye, it was not possible to derive a general solution for robustly extracting the pixels belonging to annual rings with the methods presented in that chapter due to rough condition and non-trivial shape of the tree rings. At the time of previous studies on annual ring detection and wood growth analysis discussed in Section 2.1, neural network based methods were not widely used mainly due to lack of computational power and research. Thus, it is interesting to utilize deep learning methods in solving a problem that has been studied before but not completely solved in a general case.

Neural networks, as the name suggests, consists of neurons which simulate those of the human brain [5]. These neurons have an n number of inputs denoted x_n that have weights w_n. By summing the neurons, we get the logit of the neuron: $z = \sum_{i=0}^{n} w_i x_i$, which also usually includes a bias term b. Logit is a value that can be passed to functions to get some output depending on the application and the output can be passed on to further neurons, forming a neural network. In the human brain, the neurons are organized in layers where the information moves from layer to layer so that higher-level information is in the later layers, whereas raw visual data is in the bottom layer. Neurons in deep learning are organized in a similar way: the bottom layer contains input data that are images of wood end-faces in this case, followed by layers which extract features first at a simple level then at a more detailed level and the last layer computes the final answer that is, in this case, the detected annual rings. Figure 15 shows a simple feed-forward organization of neurons in a neural network.

Convolutional neural networks [5] are type of neural networks where the network is arranged in a similar fashion as in the visual cortex of human vision, thus especially effective for computer vision applications. As its name suggests, CNNs use convolution operations in its layers, whereas regular neural networks mainly use general matrix multiplication. These convolutional layers are three-dimensional filters with a specific width, height and depth and operate by convolving the input with a kernel, which is typically smaller than the input. Filters of convolutional layers are much like those used in traditional image processing; they extract features such as edges or lines from an input and output a feature map, the difference being that they

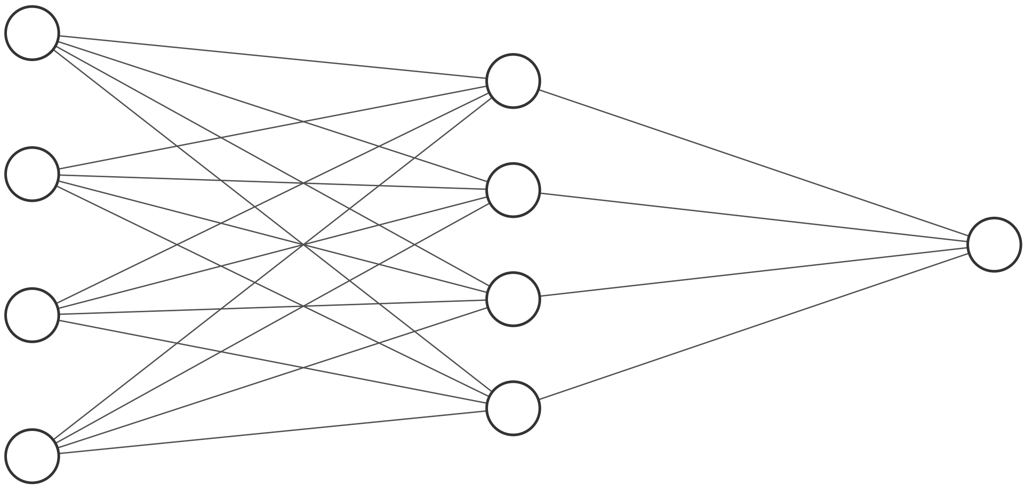


Figure 15: A simple neural network with four neurons per input and hidden layers and one neuron as output layer.

have learnable weights, which allows the filters to learn to extract desired features automatically. Common filter sizes used in CNNs are of size 3×3 or 5×5, as they are effective at capturing details and useful information, while also keeping the number of parameters small. Each convolutional layer produces a new three-dimensional block of information while combining information of previously learned features. The process is perhaps best explained via an illustration, as in Figure 16. In addition to the convolutional layer, CNNs typically include max-pooling layers after convolution layers. They condense the output feature maps from convolutional layers into smaller blocks, thus reduce the complexity of the network and sharpen the obtained features. Between the convolutions and pooling, there are non-linear activation functions that process the linear activations of convolution layers. Non-linear activation functions are important in neural networks as they decide whether each neuron should be activated or not based on its relevancy to the prediction task, thus nonlinearity is necessary in order to learn nontrivial relationships between the neurons. Activation function can, therefore, be viewed as a 'gate' between neurons of different layers. Restricted Linear Units (ReLU) are most commonly used non-linear functions in CNNs, using $f(z) = \max(0, z)$ as the activation function.

Semantic segmentation [13] is a specific application of deep learning where each pixel in an input image is labeled to a certain class. While it was difficult to isolate annual rings with traditional image segmentation techniques, as discussed in Section 3.1, semantic segmentation aims to understand and parse the scene by using the previously discussed CNNs similarly to that of human vision. CNNs are capable of segmenting the image into different classes even if they have non-trivial features, in this case capable of learning the properties of annual rings. In the recent years, semantic segmentation has surpassed traditional image segmentation methods by a large margin, and the gap continues to rise as deep learning and semantic segmentation are popular and fast-moving fields of research, with new state-of-the-art models being published frequently.

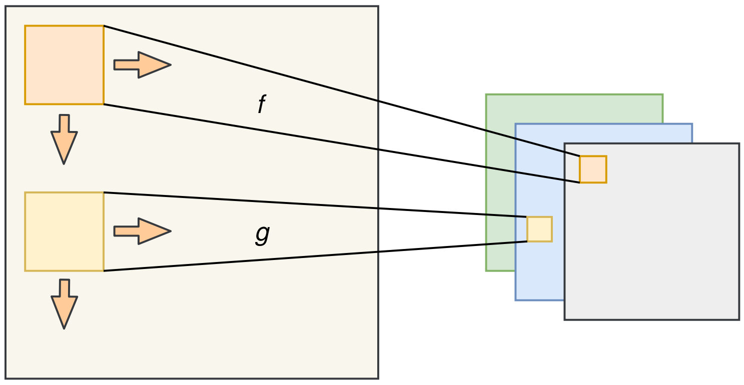


Figure 16: Illustration of a convolutional layer. Convolution between an input feature map (left) and a filter yields output feature maps (right).

CNNs for semantic segmentation consists of an encoder and a decoder part, where the encoder part learns and finds features and the decoder part maps upsamples these low-resolution features into pixel-wise predictions of classes [13]. Final layers in a semantic segmentation network are a softmax layer, which calculates the probability distribution for the classes based on the output of the neurons [5], followed by a pixel classification layer, which does the labeling of each pixel with a class. There are different choices of loss functions to use at the pixel classification layer, which are discussed in Section 3.2.3. A few common network architectures for semantic segmentation are explained in detail in Section 3.2.2, which also dives deeper into principles of semantic segmentation. Training data in semantic segmentation consists of pixel-level labeled images with a user-defined number of classes from which the CNN learns features and predicts output pixel classes.

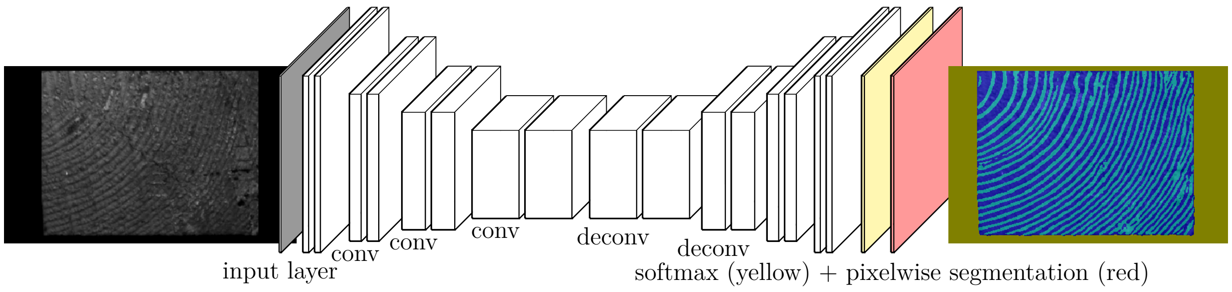


Figure 17: Illustration of a CNN performing semantic segmentation.

The remainder of this section is organized as follows: Section 3.2.1 presents and explains the dataset used in training the CNNs of this thesis, followed by Section 3.2.2, which covers a few popular network architectures and describes in detail the model used in the final implementation of this thesis. Section 3.2.3 discusses network training both in general and in the specific case of this thesis and finally, Section 3.2.4 describes postprocessing of the segmented images, and finally, Section 3.2.5 presents the results of the deep learning approach for the annual ring detection. Section 3.3 presents a recap for this chapter as a whole.

3.2.1 Dataset

Data is arguably the most important and time-consuming part of a machine learning process, especially with deep networks [13]. Semantic segmentation requires a dataset of pixel-level labeled images from which it learns to label new and unseen images. As stated earlier, there are tree rings of various shapes and sizes, and some have defects such as broken rings, knots or tar. The dataset should comprehensively cover this variability in the data in order to handle different types of wood end-faces. There were no public datasets of labeled wood end-faces, thus the dataset had to be built and pixel-level annotated from scratch. Pixel-level labeling of images is a time consuming manual labor, especially in this case, where the rings are thin and sometimes very faintly visible or broken. The images were annotated into three classes: background, woodblock and the annual rings. For the author, labeling took approximately 15 to 30 minutes per image. To overcome this tedious job, one can, for example, ask each of his colleagues to label a couple of images. There are also several paid commercial annotation services, but this thesis does not address the quality of those.

Image annotation. While in this thesis the annotation happened to be a laborious job, it is not always the case, as for an easier segmentation task, i.e., one with larger and more uniform and clear edged objects, it would be possible to do annotation semi-automatically using image segmentation methods such as watershed or k-means clustering. This paragraph presents those methods along with other tricks for semi-automatic annotation and discusses future visions of AI-assisted image annotation.

Watershed was covered in Section 3.1.2. While it was determined inefficient for the segmentation of annual rings thus could not help in the annotation process of this thesis, it could be used in an easier annotation task.

K-means clustering [2] is a popular clustering method in data mining and it can be applied to image segmentation as well. Given k number of cluster classes, k-means aims to minimize the average squared distance of points belonging to the same cluster based on local search. These k center points are chosen arbitrary, then each data point, pixel in an image in this case, is assigned to nearest of the k center points. After that, center points are recalculated based on the center point of mass of the clusters. This is repeated until stabilization. The algorithm can be run multiple times in order to find a potentially better clustering at the cost of computation time. Arthur and Vassilvitski proposed an improvement to the classic k-means algorithm, called k-means++ [2], where the cluster points are chosen with specific probabilities by using randomized seeding technique and the rest of the algorithm proceeds as in the original k-means algorithm. This resulted in a slightly increased accuracy and a substantially increased efficiency with logarithmic worst-case time complexity.

In the data of this thesis, k-means clustering using the k-means++ method was capable of partly segmenting the best condition wood end-faces, however there are still many flaws to manually fix that it is practically not much faster than annotation by hand from scratch. For an annotation task with more clear objects such as scene parsing, this method would suit well for semi-automatic labeling, especially

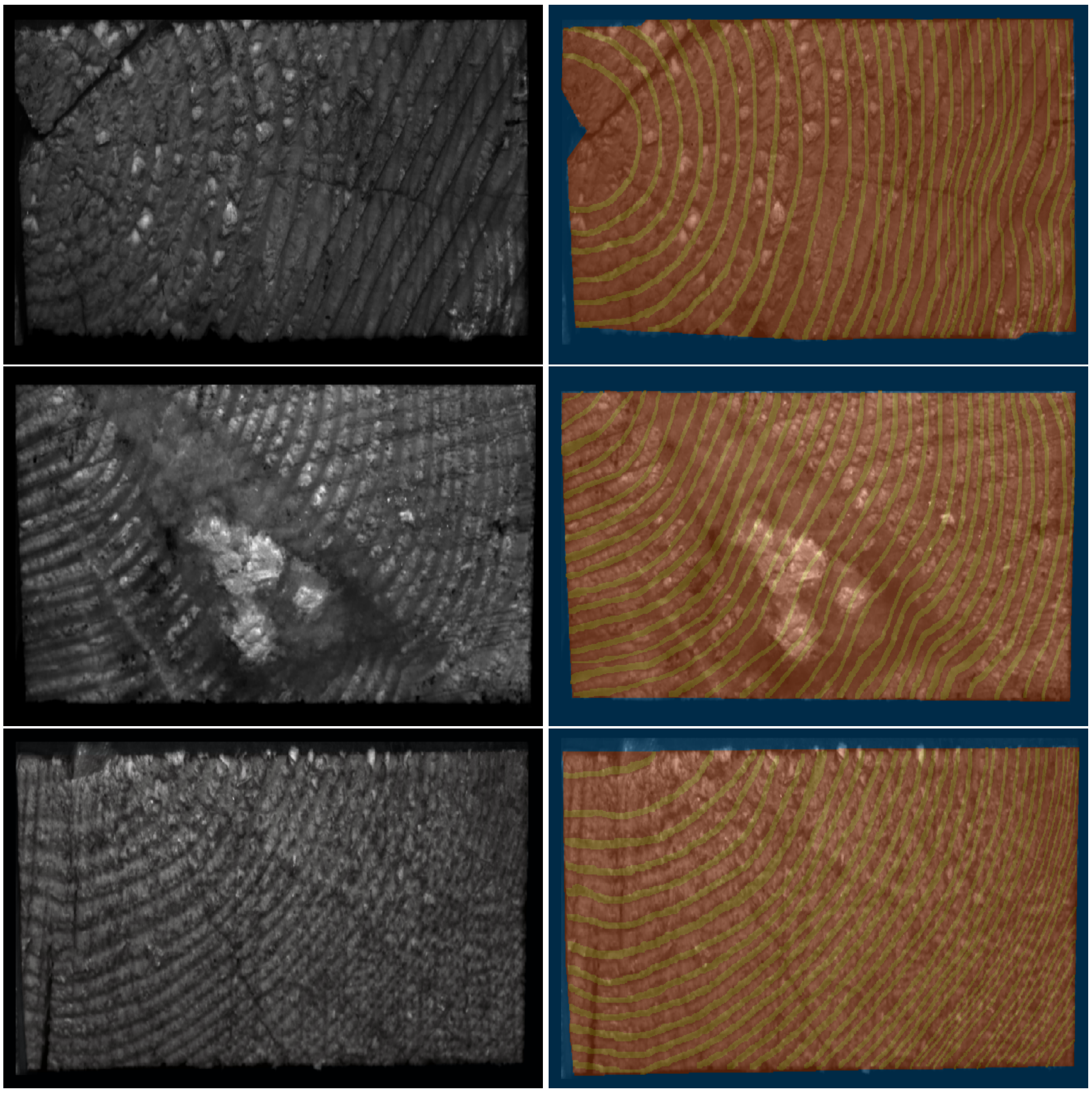
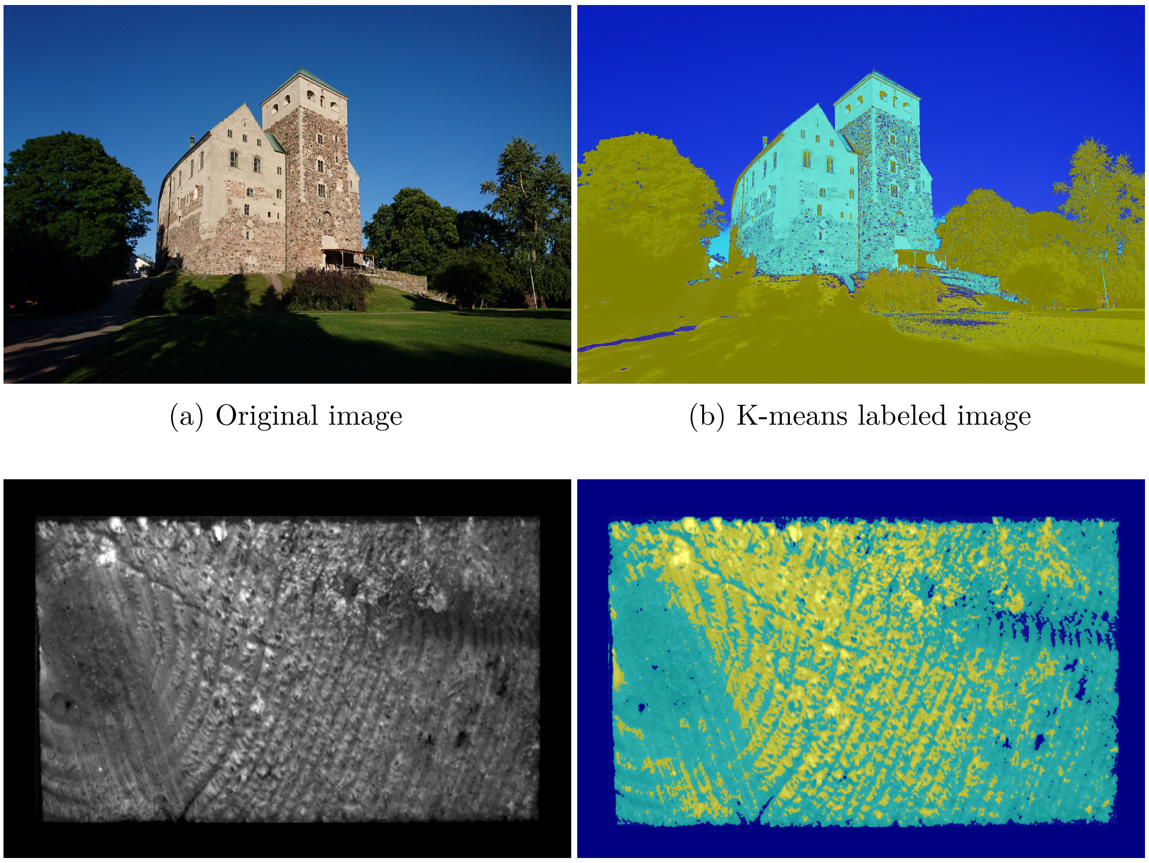


Figure 18: Examples of images (left) and their corresponding pixel labels (right).

given its speed and simplicity. An example of using the k-means method for image segmentation on an image of a scene and on a fairly good condition training image of this thesis is shown in Figure 19.

There are also a few user-friendly and ready-to-use semi-automatic annotation tools that can potentially speed up the annotation process substantially. Perhaps the most promising of them is ByLabel [26], which is a novel tool for semi-automatic image pixel annotation publicly released in 2018, which was shown to outperform previous state-of-the-art tools in all aspects and claims to be a user-friendly and fast tool. As opposed to most competitor tools where the user has to click boundary points by hand, ByLabel automatically generates boundary proposals, from which the user can choose the best match. This automatic boundary detection works by finding the edges features of the image and then splitting them based on turning angles



(a) Original image (b) K-means labeled image

(c) Original image (d) K-means labeled image

Figure 19: Example of k-means segmentation on a scene image and on a wood end-face image.

or manually by the user. This categorizes objects into three types: simple objects with one contiguous region or closed boundary, objects with holes that have nested boundaries or contiguous regions, objects divided by occlusion determined by several regions or boundaries. After that, user input is required for interactively annotating and fixing the boundaries and for choosing the best match. More specifically, the edge detection in ByLabel is done with Edge Drawing, a method from Tokal and Akinlar's 2012 paper capable of producing clean edge segments of one pixel width. It is resulting edge maps are somewhat similar to that of the Canny edge detector presented in Section 3.1.2. Based on images of example annotations in the original ByLabel paper, it seems to do a decent job even on some difficult shapes; however, all of those test images have edges clearly visible, unlike annual rings, which can be rather faint and unclear.

The neural network itself can also be used as a semi-automatic labeling tool. Once it has learned to somewhat segment new images, it can be used to speed up the annotation process: the unlabeled images are fed to the network for segmentation, then the remaining flaws are fixed by hand, making it a somewhat semi-automatic process. While this helped a little in the annotation process of this thesis, it still remained a tedious job.

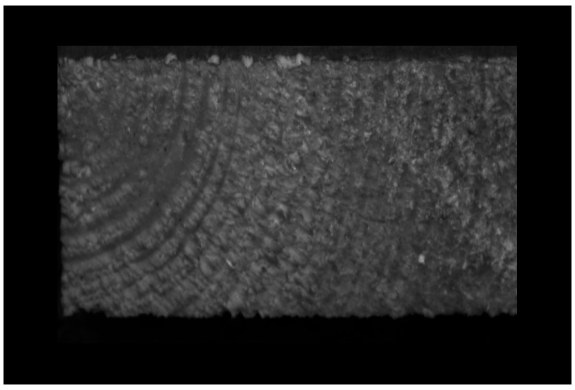
Robust semi-automatic annotation still remains an unsolved problem, however AI-assisted methods have been studied in the recent years and they could become widely used in the near future. Very recently (October 2019) an AI-assisted semi-automatic annotation tool was published for everyone as a public beta in Supervisely [11], a web platform for computer vision and semantic segmentation. According to Deep Systems, the company behind Supervisely, this is the first publicly available AI-assisted annotation tool. It is exciting to see what kind of solutions for semi-automatic image annotation will appear in the near future, as that will drastically speed up the creation of deep learning based computer vision applications and make the process more pleasant for the engineers.

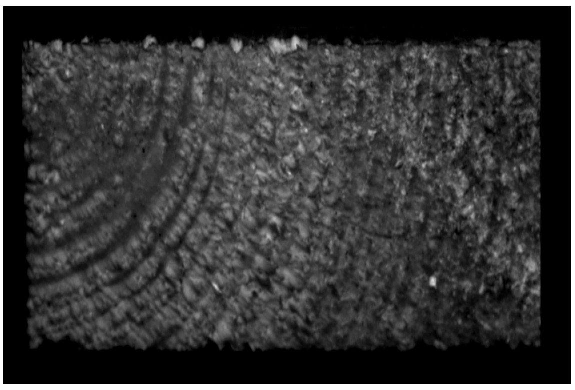
Data augmentation. After the previously discussed annotation process, the dataset consisted of 100 images and their corresponding pixel labels, however neural networks rely on a large amount of data to avoid overfitting on the training data and 100 training images may not be enough for the neural network to learn the rather difficult properties of annual rings sufficiently well without overfitting on the data. Overfitting [5] refers to the problem of the neural network not generalizing well on the data: it is perfectly fitting and learning the training data but performing poorly on new data. This is one of the most common challenges in machine learning, especially when dealing with complex and deep networks such as in this thesis. Luckily, new training data can also be created artificially by using data augmentation techniques [29]. Dataset is inflated by warping the training images with affine transformations and color transformations such that geometric operations are also performed on the corresponding pixel labels. Augmentation is based on the fact that even though a human can see that an image after zooming and rotating is still the same, a neural network cannot. Geometric image manipulations to apply on the training images can be, for example, flipping along x or y–axis, translation, cropping, skewing, or rotation. Cosmetic transformations can be applied by simple kernel filters. Depending on the data and use-case, these could include, for example, blurring, sharpening, or contrast enhancement. While augmentation is proven to be efficient in fighting overfitting, it should still be used on a reasonable scale; inflating a small dataset into a really massive one by combining multiple transformations may result in further overfitting. However, within reasonable limits, data augmentation will most likely improve the network substantially. Besides trial and error, it is possible to use search algorithms for finding an optimal amount of data augmentation.

In addition to data augmentation, there are other countermeasures against overfitting. These methods are based on modifying the neural network architecture and are presented in Section 3.2.2, which discusses the network architecture.

Data augmentation should reflect the problem, therefore only type of augmentation that could realistically appear in the unseen data was done, thus the following offline data augmentation was done on the dataset of this thesis:

- Reflection over y–axis with a probability of 0.95
- Reflection over x–axis with a probability of 0.05. Low probability because woodblocks usually lie in the factory line so that the pith is at the left side
- Random zooming of factor 0.95 to 1.05 with a probability of 0.98
- Random rotation of -1 to 1 degrees with a probability of 0.5. The wood is straightened algorithmically as explained in Section 2.4, therefore it does not make sense to rotate the woodblocks much
- Random contrast modification with a probability of 1 within a small range, as the contrast does not vary a lot in the data
- Random brightness modification with a probability of 1 within a small range, as brightness does not vary a lot in the data
- Random blurring with a probability of 0.05 with a Gaussian blur of size 3. Small amount of images are a bit blurry due to the factory line vibrating.


(a) Original image


(b) Example augmentation

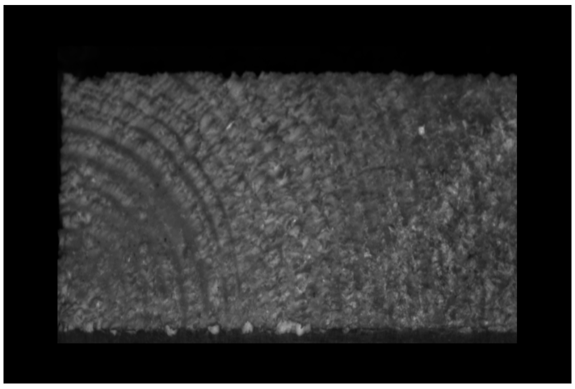

(c) Example augmentation

Figure 20: An original training image and corresponding augmented images.

Additionally, online augmentation with a slight amount of random zooming, translation, and rotation was performed. Online augmentation means augmenting training data after each epoch during training, which essentially prevents the network from seeing the same images every epoch, which is a full cycle over the training set. This further reduces the risk of overfitting on the training data at the cost of increased training time.

The dataset was divided into training data and validation data [5] with a split of 90/10. It is tempting not to use validation data, as the dataset was built through hard and time-consuming work. However, validation data plays a vital role as it enables a fair evaluation of a network model, and it is crucial for spotting overfitting. Validation data is used to monitor the training process: when the validation loss starts increasing, the network is starting to overfit on the training data and stops learning any useful features. In other words, validation data tells us how the network behaves on new, unseen data. It also gives us an estimation of the accuracy of the network on new data, making it an important tool in tuning hyperparameters, such as learning rate or minibatch size. In addition to training and validation datasets, an external testing dataset was created, which is used to give an unbiased evaluation of the model after training. The testing dataset consisted of 18 labeled images, that were chosen so that they represent the variability of the data. Network training and hyperparameter tuning are explained further in Section 3.2.3.

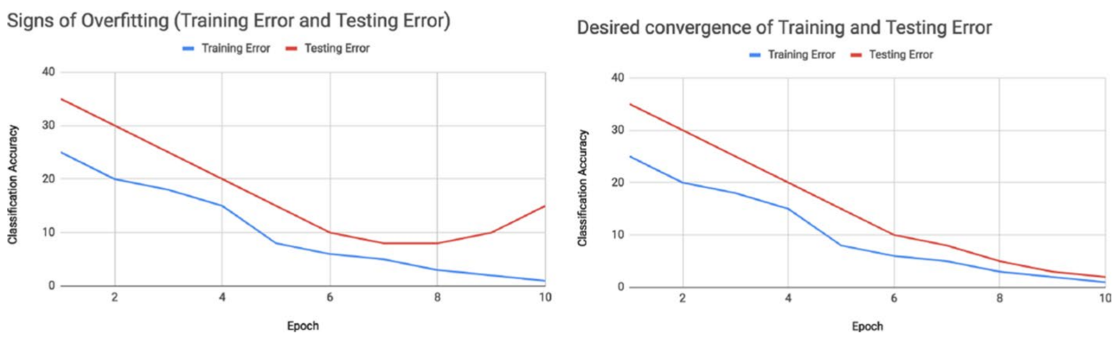


Figure 21: Image from Shorten and Khosgoftaar's survey [29] visualizes overfitting. Plot on the left shows converge point where validation loss starts to increase while training loss is still increasing, which is a sign of overfitting. Plot on the right shows a desired situation.

3.2.2 Network Architecture

Basics of convolutional neural networks and deep learning were covered in Section 3.2. This subsection presents in detail the semantic segmentation CNN architectures used for the annual ring detection task of this thesis.

U-Net. U-Net [27] was chosen as a starting point for the final implementation due to its simplicity, speed, and capability of getting good results even on small data-sets. It was originally designed for Biomedical Image Segmentation, but it has also proven to be decent at other types of segmentation tasks such as retinal blood vessel segmentation and satellite image segmentation. The application of annual ring detection is a somewhat similar task as well, further motivating the choice of U-Net as a good starting point on building a neural network for the problem of this thesis. An adaptation of U-Net was also successfully used as the network architecture in Fabijanska and Danek's DeepDendro tree rings detector [12]. The typical use of CNN's had been on classification problems, which usually have a training set of thousands of images. In contrast, biomedical image segmentation requires precise labeling of each pixel and does not have large annotated training sets available, which is also the case in the segmentation task of this thesis. The motivation behind the authors of U-Net was to overcome this type of problem. Released in 2015, U-Net is still a popular choice in the semantic segmentation of medical images and similar tasks.

U-Net is built upon the idea of Fully Convolutional Network (FCN) [19], which is a type of neural network that extends convolutional neural networks to input of any size and produce an output of the same size. FCNs are trained end-to-end, which means that learning by backpropagation and interference by feedforward computation is done on the whole image. Architecture of FCN is visualized in Figure 22. U-Net [27] was modified to give more precise segmentations with fewer training images compared to the FCN architecture. U-Net has a symmetrical expansive (encoder) and contracting (decoder) path, which forms a u-shaped organization of layers, as shown in Figure 23, where U-Net gets its name. Each level in the encoder part consists of two 3×3 convolutions, a ReLU and a 2×2 max-pooling operation. Stride size 2 downsamplings performed by max-pooling operation doubles the number of feature channels. In the decoder part, each level halves the feature channels by upsampling with 2×2 convolution, followed by two 3×3 convolutions and a ReLU, symmetrical to the encoder part. There are skip connections between the upsampling and downsampling paths, which combine the deep feature maps from the decoder part with the shallow and fine-grained feature maps of the encoder part. This allows effective recovering of fine-grained details, necessary for medical image segmentation tasks as well as the tree ring segmentation task of this thesis. The final output layer maps each of the feature vectors into user-defined number of pixel label classes. The architecture of Figure 23 consists of 23 layers, and has a depth (i.e. number of downsamplings) of four but the depth can be increased or decreased based on the task.

The U-Net model built for this thesis opted for a depth of three, the size of largest filter therefore being 512, which upon testing proved to be the best trade-off between performance and complexity. Increasing the depth to four did not cause a meaningful increase in accuracy; thus, it would have brought unnecessary complexity and slower interference time, whereas lowering the depth to two caused a clear decrease in accuracy.

Separation of touching objects is a common problem in medical segmentation tasks, therefore weights of the loss function were altered in the U-Net paper by giving the background labels a larger weight to prevent objects of the same class from touching each other. Similar modifications were also needed in this thesis

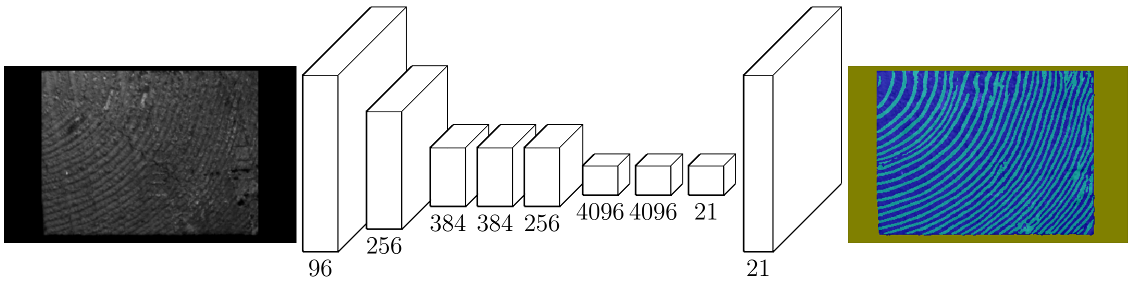


Figure 22: Fully Convolutional Network end-to-end architecture from [19], where forward direction is interference and backward direction is learning.

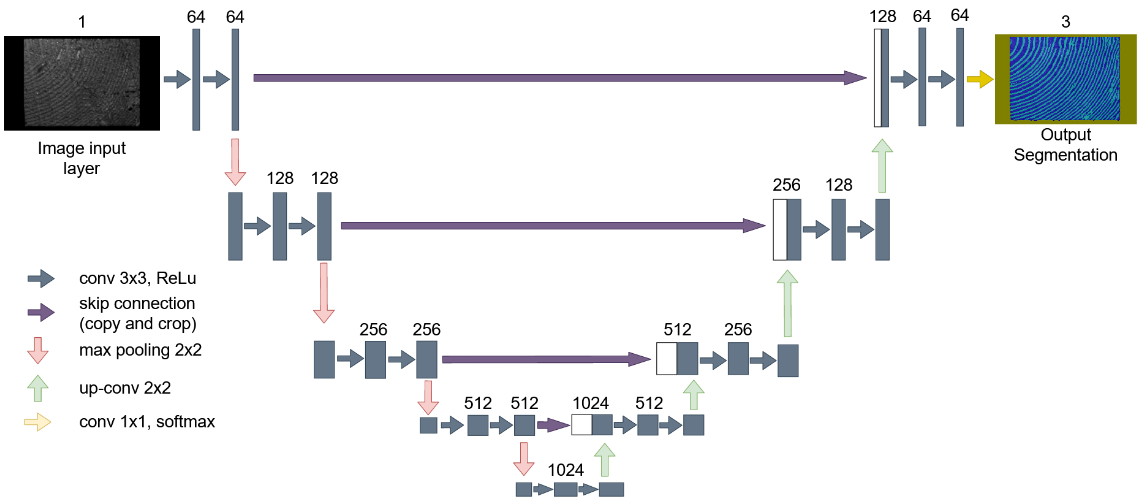


Figure 23: U-Net [27] architecture.

due to the same problem of background labels being dominant. Class balancing is explained further in the Section 3.2.3, which discusses network training. Another shared problem between the medical image segmentation task in the original U-Net paper and the wood end-face segmentation task of this thesis is having a small amount of training data available. Ronneberger, et al. used data augmentation to artificially expand the data-set by performing elastic transformations, which allowed the network to see many variations of the data and substantially improved results. Data augmentation techniques performed in the dataset of this thesis were described in Section 3.2.1.

DenseNet [17] is a CNN architecture that contains short and dense connections between layers, so that layers of same feature-map sizes are directly connected. A dense block is visualized in Figure 24. This encourages feature reuse and improves information flow between the early and the deeper layers without increasing the number of parameters. Using DenseNet inspired dense blocks in place of the convolutional blocks in the U-Net model resulted in a slightly increased performance in the application of this thesis.

Even after the augmentation process discussed in Section 3.2.1, the dataset is still relatively small. Therefore, there is a risk of overfitting on the data. Modifications for preventing overfitting were done on the U-Net derived network architecture of this thesis: transfer learning and added regularization methods.

The process of human learning builds on the previously learned; for example, it is considerably easier to learn the theory of deep learning if one has already acquired knowledge on linear algebra and probability theory. This is true for machine learning as well: training a deep neural network from scratch with uninitialized weights on a small dataset such as that of this thesis takes a long time and can easily lead to overfitting thus the network might not generalize well on the data, however by using weights from a pre-trained network and by transferring this knowledge on a new task, it is possible for the deep network to learn faster and easier. This technique is called transfer learning [13], and it has proven to improve performance compared to randomly initialized weights in initial layers even if the transferred task does not resemble that of the new task. This is because the features learned in early layers of convolutional neural networks are generic features such as edges, lines and shapes, which are needed in all feature extraction tasks, including the annual ring detection task. As explained earlier, the later layers are capturing more specific and higher-level features, therefore transfer learning should only be used on the initial layers, especially if the tasks differ a lot. There is a great variety of publicly available large-scale datasets and network architectures to choose from for the transfer learning process. There are also publicly available network models that have already been trained on large datasets, which makes it easy to do the transfer learning: download the pre-trained model and replace the initial layers of the new task with those of the pre-trained network, given that the size of the copied layers matches the replaced ones. For the network of this thesis, weights of initial layers from a pre-trained VGG-19 network [30] were used. VGG is a very deep CNN with small convolutional filters designed for large-scale image classification tasks. Its architecture consists of a stack of convolutional layers followed by three fully connected layers and a softmax layer. A different number of convolutional layers were tested, and it was determined that VGG-19 with 19 convolutional layers was the best performing of these VGG variants. Simonyan and Zisserman had found that adding more layers than 19 did not increase the accuracy substantially while having fewer layers did not perform as well. The particular VGG-19 network used in the transfer learning task of this thesis was trained on a million images at the ImageNet dataset, which contains images of 1000 classes, thus has learned a rich representation of low-level features. It is a popular model and publicly available, for example, at the official Github repository of ONNX, which is an open ecosystem for sharing neural network models. ONNX models share a common format of .onnx, which can be exported into any major neural network framework such as TensorFlow and Keras. Using transfer learning proved to make the training process of the annual ring detection network faster and more robust with less oscillation.

Dropout [5] is a popular regularization method that decides whether or not a neuron is kept active or set to zero by some user-defined probability. This ensures that the neural network does not become too dependant on some combination of neurons and therefore reduces the risk of overfitting and can help the model to generalize better. Adding dropout layers after each convolution layer improved the neural network of DeepDendro [12], and likewise in the CNN of this thesis, it allowed training the network for a longer time before overfitting on the data thus improving the performance of the model.

Batch normalization [5] is another regularization method which can accelerate the training process, allowing the model to generalize faster. It operates by normalizing the inputs of each layer in a neural network, preventing shifts in the input distributions and acting as regularization, thus removing the need for the previously discussed dropout layers. Batch normalization was also used in the densely-connected blocks of DenseNet [17] and added to the U-Net model with dense blocks of this thesis.

Two different U-Net models were implemented for this thesis using the previously discussed modifications: U-Net with pre-trained VGG-19 encoder, dense blocks and added batch normalization, U-Net with pre-trained VGG-19 encoder and added dropout layers. Architectures of these U-Net variations are shown in Figure 24 and 25, respectively.

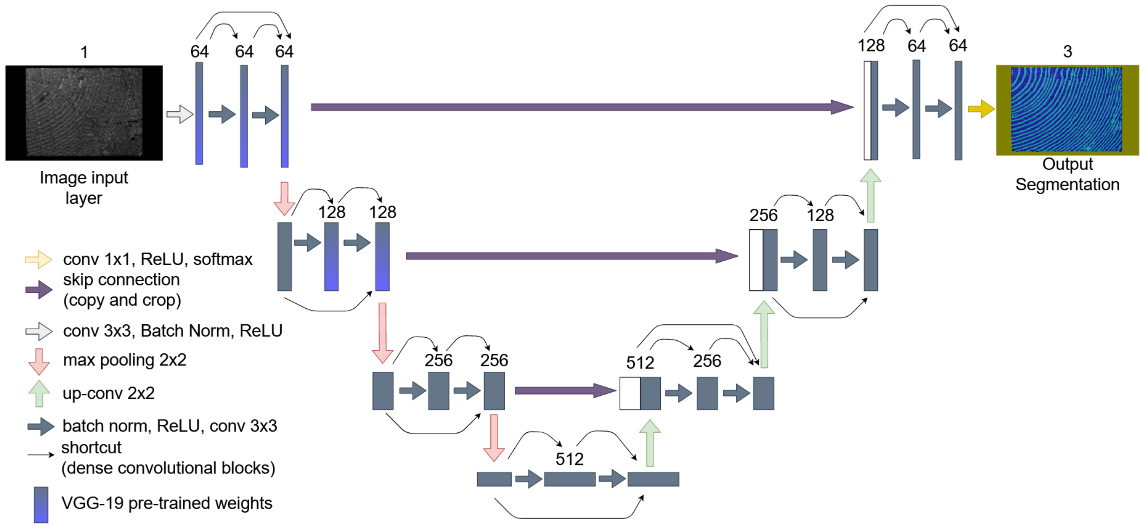

1
Image input layer
64
64
64
128
128
256
256
512
512
256
512
256
128
128
64
64
3
Output Segmentation
conv 1x1, ReLU, softmax
skip connection (copy and crop)
conv 3x3, Batch Norm, ReLU
max pooling 2x2
up-conv 2x2
batch norm, ReLU, conv 3x3
shortcut (dense convolutional blocks)
VGG-19 pre-trained weights

Figure 24: U-Net [27] with dense convolutional blocks, batch normalization and pre-trained weights on initial layers.

Other networks. Other base network architectures that were implemented and tested for this thesis include SegNet and DeepLabV3+, but they did not provide as good results as U-Net on this particular task. These network types are briefly introduced next for the sake of comparison and in order to highlight the differences between these widely used network models.

SegNet [3] is another encoder-decoder type architecture for semantic pixel-wise segmentation of images. It was published in 2016 with a primary goal of scene understanding applications, thus designed to be computationally efficient with less trainable parameters than its competition. Its encoder part is identical to that of the VGG-16 network's first 13 convolutional layers. Similarly to transfer learning on U-Net, as described earlier in this section, SegNet uses pre-trained weights from VGG-16 that was trained on a large dataset. In the decoder part, SegNet does upsampling

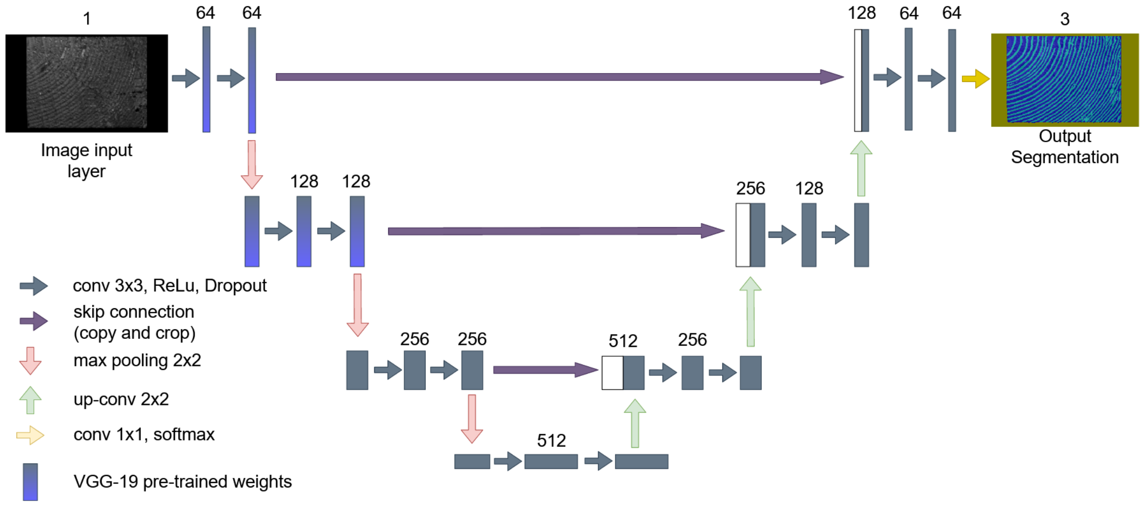


Figure 25: U-Net [27] with dropouts and pre-trained weights on initial layers.

without learning by using max-pooling indices, then convolves the feature maps with a trainable filter. Batch normalization is applied to each of these feature maps in both the encoder and decoder parts. The architecture is similar to U-Net, but it reuses pooling indices instead of transferring the features directly in order to save memory, and it does not have skip connections for recovering fine details. Final decoder layer containing high dimensional features outputs to a softmax layer, which classifies each pixel with probabilities for the desired number of classes. The decoder part is symmetrical to the encoder, therefore it also contains 13 layers plus the softmax layer. Hence, the total depth of the original SegNet network is 27 layers. The architecture is visualized in Figure 26. Tested on well-known scene parsing datasets such as CamVid, SegNet proved to perform competitively, especially performance-wise, at the time of its publishing in 2016. As of now, newer architectures have reached higher performances, although they are also often more complex.

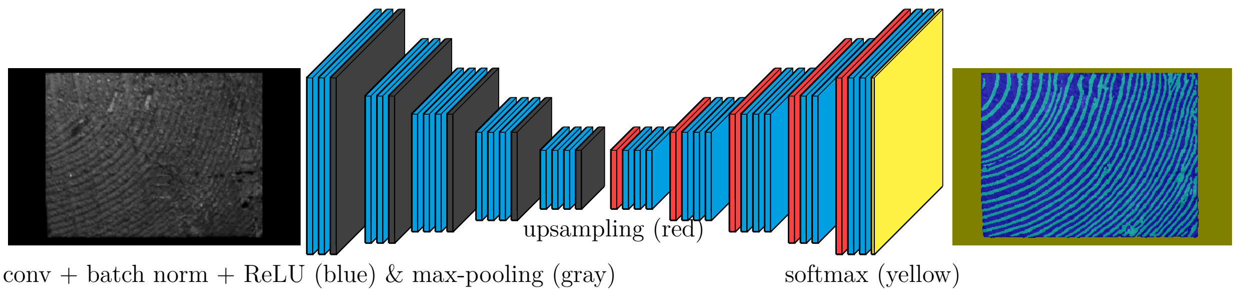


Figure 26: SegNet [3] encoder-decoder architecture.

DeepLabV3+ [7] is a semantic segmentation architecture developed by Google. It is an improvement over its predecessor, the DeepLabV3 architecture, which main idea is to apply parallel atrous convolutions of different rates in order to capture information and features at different scales. This method is called the Atrous Spatial Pyramid Pooling (ASPP). Such models that use ASPP are used to extract dense feature maps to capture rich contextual information through pooling, but that requires the final blocks to be dilated, which is computationally expensive. Encoder-decoder type networks such as previously described U-Net on the other hand are faster to compute as they do not require features to be dilated and they recover object boundaries accurately. DeepLabV3+ tries to combine the benefits of both these methods, thus improving the original architecture. This is done by adding a decoder part for recovering object boundaries to the DeepLabV3 architecture while keeping the atrous convolutions in the encoder part in order to extract more dense features. In other words, DeepLabV3+ features an encoder-decoder structure with an efficient encoder part from the DeepLabV3. The original paper used a pre-trained Xception network as its contracting part but it is possible to experiment with other pre-trained networks as well. Xception uses depth-wise separable convolutions which deal with depth dimensions in addition to spatial dimensions: a depth-wise and a pointwise convolution is performed at each convolution step. Networks utilizing depth-wise convolution should reduce the computational costs without sacrificing performance. This has been a popular choice in many recent network architectures. High-level architecture of the DeepLabV3+ network is shown in Figure 27. At the time of its release in 2018, DeepLabV3+ was the state-of-the-art on popular PASCAL VOC 2012 and Cityscape datasets. For this thesis, two variants of DeepLabV3+ were implemented: one with Xception encoder and other with MobileNetV2 encoder.

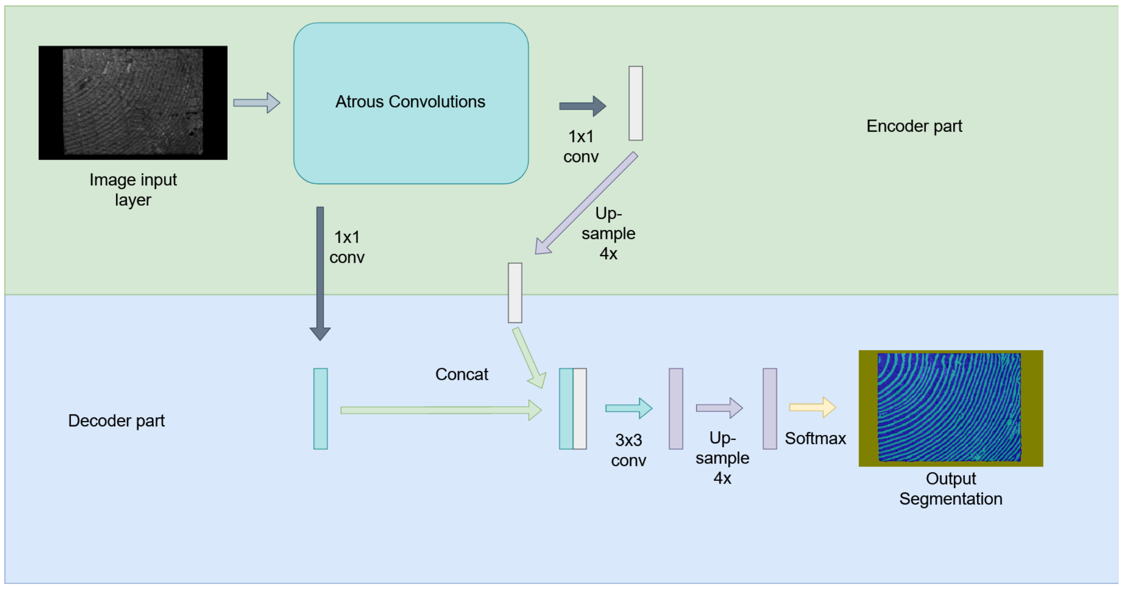


Figure 27: DeepLabV3+ [7] high-level architecture.

Network architectures implemented in this thesis are evaluated and compared in at Section 3.2.5.

An introduction to parameter tuning in deep learning neural network training and an in-detail explanations of training options used in the U-Net models of this thesis are presented in the following Section 3.2.3.

3.2.3 Network Training

The workflow of training a deep learning model is visualized in an easy to follow manner in the diagram of Figure 28, derived from Buduma and Locascio's book [5]. It shows the main steps of building a neural network model for any deep learning application. The training process consists of monitoring metrics and making choices accordingly. These metrics are typically accuracy and loss for both the minibatch currently in training and validation set. If after an epoch of training the loss is decreasing, and accuracy is increasing for both the minibatch and the validation set, it is safe to continue training for another epoch, whereas if the validation loss starts increasing, the network has started overfitting on the data and the training should be stopped. Minibatch is a subset of user-defined size from the training data that is used at each iteration, and an epoch is a full-cycle over the training data. In some cases, the network may continue to improve on the training data, but it is overfitting on the validation data, that is why it is important also to monitor the metrics on validation data. The importance of the validation set was discussed in more detail in Section 3.2.1.

The most important parts of a deep learning model are the dataset and network architecture, discussed in Section 3.2.1 and 3.2.2, respectively. If the data is comprehensive and well-defined and the neural network architecture suits the task, then the network should learn from the data and perform well on training data and, most importantly, on the test data. On the other hand, if there is not enough data or if the data is not sufficiently good, then the network might not perform well on the training data or could overfit on it.

What the diagram of Figure 28 does not take into account is choosing training options and doing parameter tuning, which can further improve the model performance and allow training for a longer epoch. Furthermore, poorly chosen options could, in the worst case, cause the whole learning to be unsuccessful; therefore, the machine learning engineer should know the basics behind the learning process. Luckily, optimization in machine learning is not as difficult as optimization in a general case. The object in deep learning training optimization is to find parameters that best reduce a cost function, which is typically a performance measure of the training and validation [15]. This section explains standard parameters in the network training, then discusses the training options used in the training of the best performing network for this thesis.

Learning and optimization. Learning rate, usually denoted by ϵ, is an essential hyperparameter in a machine learning training process [5] as it determines the size of the steps to take towards the local minimum after each training iteration. Picking a too large value for the learning rate may cause the network to diverge away from the real minimum, and picking a too-small value, on the other hand, causes the training to be slow. The task of finding the local minimum in neural network training is an optimization problem for minimizing squared error over training data by updating the values of weights w of the neurons. There are various algorithms for solving this optimization problem, called solvers or optimization algorithms. The optimal choice of learning rate varies based on the solver, network architecture and the application but a common strategy for finding a suitable learning rate is to start with a low value and continue increasing it by a magnitude of 10 until the largest value that does not cause divergence has been found. It is also possible to start with a larger value and then lower the learning rate before the result starts to diverge away from the correct path. Two commonly used optimization algorithms were tested for the neural network training of this thesis: the Stochastic Gradient Descent (SGD) and Adam.

SGD [15] is a popular choice for both machine learning and deep learning applications. It operates by calculating the average gradients of minibatches controlled by the learning rate; in other words, it takes steps towards the direction of the negative gradient to minimize the loss function. SGD tends to be slow and oscillating as it is, however it can be improved by adding a momentum term which accelerates the training by observing previous gradients steps and having them contribute to the next steps. This contribution is essentially a moving average of past few gradients of which effect can be controlled by a scalar.

Adam [5, 15] is an adaptive type of optimization algorithm, meaning that the momentum is chosen dynamically by estimating first-order moments of the gradient, thus potentially making it easier to control as there are fewer hyperparameters to tune by hand. Adam features an exponentially weighted and decaying moving average is maintained in order to ignore old gradients, and an added bias correction factor applied to the moments. At the moment, there is no correct answer on the choice of the best optimization algorithm for a deep learning problem, although adaptive learning rate algorithms such as the previously described Adam have been proven to often perform robustly. The choice, however, seems to be mostly up to the engineer's familiarity with the algorithm, as to make parameter tuning easier [15]. It is also worth noticing that breakthroughs in the field of deep learning are most often acquired through discovering and inventing novel network architectures and rarely by fiddling with optimization algorithms [5], and in practice it might also make sense not to spend too much time on improving and choosing the solver but preferably on improving the network and data. Upon initial testing of both SGD and Adam optimizers it was noticed that Adam felt slightly easier to control in this case, thus it was chosen for the optimization algorithm of the network training task of this thesis.

Minibatch. Minibatch size [15], as mentioned before, defines how many training samples are used to update the neuron weights and to take steps towards the minimum at each iteration, therefore also defines the number of iterations in each epoch. The choice of minibatch size thus affects the speed of the training process, as there are fewer total iterations in each epoch, the larger the minibatch size is. The use of a small minibatch size might result in better regularization due to noise caused by variability in the data, but the training can oscillate and be more difficult and time-consuming. When choosing minibatches, it should be ensured that the data in minibatches is picked at random, especially if the dataset is organized in a biased way such that similar training images are arranged successively. This also makes sure that there is a different ordering of data in each epoch, which reduces the risk of overfitting. Alternatively, the dataset itself can be shuffled before each epoch, which ensures non-biased ordering of the data.

Maximum minibatch size is restricted by the available hardware, GPU memory can fit a limited number of training images on a single iteration, and the time it takes for training a single iteration is dictated by the GPU's computational power in addition to the depth and complexity of the network model. Furthermore, a minibatch size of power of 2 can generally achieve a faster runtime on GPUs, as they are optimized for parallel calculations with power of 2 size arrays. Training can also be done on a CPU, which might have a larger memory depending on the setup and could, in that case, fit a larger minibatch size, but generally operates much slower than a GPU. Just in the recent years, it has become practical to train deep networks, which are computationally demanding tasks, thanks to the rise of computational power, which still follows Moore's law and continues to increase. A rather modest Nvidia GeForce GTX950 GPU from 2015 was used in this thesis, and training the neural network on said GPU for 100 epoch with minibatch size of 4 took approximately 12 hours. On a typical early 2000's computer, this would have taken days, and with 90's hardware, the task could not have been done at all. While the GPU of this thesis was sufficient for training several prototype level networks on a quite small dataset, more powerful hardware or cloud computing will be considered when moving forward and enlarging the dataset in the project. It is not practical to train for such a long time, and results of larger minibatch sizes could not be tested due to limitations in the GPU memory.

Loss functions. In pixel classification tasks, loss function determines the assignment of class labels for each pixel, and thus, in this case, it is the last gatekeeper between the neural network and the real-world results of detecting the annual rings. Two loss functions [31], Weighted Cross-Entropy (WCE) loss, and Generalized Dice Loss (GDL) were implemented and compared for the neural network of this thesis as they are two of the most commonly used.

Entropy [7], in this context, is an information-theoretic measure for describing uncertainty in a probability distribution fist coined by Claude Shannon. Cross-entropy measures the probability error in case of multiple classes that are not mutually exclusive. These probabilities are acquired by a softmax layer, which calculates the probability distribution for the classes. Problem with cross-entropy loss is that it addresses each pixel independently and equally, therefore if the classes in a semantic segmentation task are in imbalance, it may be biased towards the dominant classes, especially if the distribution is severely uneven [27]. In this case, the number of pixels belonging to annual rings is substantially smaller than the number of pixels belonging to background and woodblock. Therefore class distribution and balancing should be taken into account. There are a couple of ways to do class balancing: class weighting by median frequency, inverse frequency, or average frequency. Weighted cross-entropy with median balancing performed best in this case and improved the segmentation substantially. It is done by calculating the frequency of pixels in each class by dividing class pixel count by total image pixel count and then for each class by calculating the median frequency by taking the median of the frequency and dividing it by the frequency. Weighted Cross-Entropy is then defined as [27]

$$E = \sum_{x \in \mathbb{Z}^2} w(x) \log(p_{l(x)}(x)), \tag{10}$$

where x is a pixel in the image, w is a weight map which balances the classes and p_l is the probability of label l given by the softmax layer.

Generalized Dice Loss [31] evaluates segmentation of multiple classes using a single score based on measure of overlap and alleviates the problem of class imbalance. Calculating loss L between image I and ground truth T for K classes using GDL is defined as:

$$L = 1 - \frac{2 \sum_{k=1}^{K} w_k \sum_{m=1}^{M} I_{km} T_{km}}{\sum_{k=1}^{K} w_k \sum_{m=1}^{M} I_{km}^2 + T_{km}^2}, \tag{11}$$

where M is the number of elements of I and w_k is a weighting factor defined as the inverse area of expected ground truth region:

$$w_k = \frac{1}{(\sum_{m=1}^{M} T_{km})^2}. \tag{12}$$

Both the weighted cross-entropy loss and GDL proved to significantly reduce the initial problem of class imbalance in comparison to using cross-entropy loss without class weighting, which failed to detect many annual rings as it favored the background pixels. Using WCE loss was found to behave more robustly than GDL, which caused the error loss to oscillate in this particular application of annual ring detection. GDL is considered to behave more robustly in situations where the class imbalance is great [31], whereas, in this situation, the class imbalance clearly exists in all images but is not that large.

There are also several other well-established loss functions to use in semantic segmentation, such as focal loss and Tversky loss. Additionally, it is possible to combine results from multiple loss functions. These were not implemented in this thesis due to time constraints but should be tested in future work in order to gain potential improvement to the segmentation results.

Summary. As a summary of network training, sensibly choosing the training options is important to ensure successful training. Further optimization of the parameters can slightly increase the model to obtain a couple of percentages higher validation accuracy, for example. However, the main building blocks of creating a decent deep learning model are the dataset and the network architecture. At times, neural network training can be more of an art than science as networks tend to behave somewhat differently depending on the application, and there is no correct answer as to which options and parameters should be chosen for a specific deep learning task at hand. Making small incremental changes based on findings through trial and error is a good strategy for building a successful network, however knowing the fundamental working principles of deep learning and exploring research on similar applications is essential in order to make educated guesses and modifications to the model and data.

For the U-Net based networks implemented for this thesis, as discussed in Section 3.2.2, choosing the following training parameters obtained the best results:

- Classes were balanced with median balancing in order to avoid background pixels being dominant
- Cross-entropy loss was used as the loss function in the pixel classification layer, it was found to be more stable than Dice loss
- Mini-batch size of 3 was chosen. Higher values could not be tested due to limitations in hardware (GPU memory)
- The network was trained for a total 150 epoch until accuracy and loss reached stability. Adam optimizer with a learning rate of 0.001 was used for the first 100 epoch, then learning rate was reduced to 0.0001 for the last 50 epoch as it was found that the network would overfit after 100 epoch without decreasing the learning rate.

The architecture of this network was described in Section 3.2.2 and its performance is reviewed in Table 2 at Section 3.2.5.

3.2.4 Postprocessing

Turning the output segmentation class probability map into a binary image with annual ring pixels in white and background in black was done by simply turning each pixel into the most confident class. More picky thresholding could be considered such that only the strongest responses for annual ring pixels from the network are chosen, which should reduce the amount of falsely detected annual ring pixels in the binary image however it could also ignore pixels that actually belong to annual rings, thus the thresholding should be implemented with care. Fabijanska and Danek [12] used a mean based thresholding algorithm and local contrast enhancement to extract the highest probability pixels in their neural network post-processing, which could be viable in this case as well.

For postprocessing of the resulting binary images of annual rings after the semantic segmentation, similar methods as described in Section 3.1.3 can be used. Morphological closing was applied on the output binary images of the semantic segmentation. With a small structuring element, it was successful at removing noise while preserving the desired pixels belonging to annual rings.

CNNs are balancing between using local fine-grained and global low-level information in the segmentation, often struggling to maintain spatial accuracy. Use of conditional Random Fields (CRF) [13] is a common post-processing step that can enhance this trade-off between low-level and high-level information and improve the output of the segmentation. CRF can improve the capturing of fine detail by combining low-level information with the output of the multi-class pixel classification. CRF was not used in any of the networks implemented in this thesis, but could be tested in future work in order to potentially improve the output of the segmentation.

3.2.5 Evaluation

The success of a semantic segmentation is determined by the success of the end-application [8], annual ring detection in this case. Computer vision applications are non-trivial, blindly looking at a single metric such as accuracy might not give the full answer on the success of the segmentation. Therefore, various metrics as well as a visual inspection is needed in order to evaluate the performance of neural network models on a specific application.

Table 2 showcases evaluation metrics for the CNN architectures implemented in Section 3.2.2, tested against a dataset of 18 labeled images, which were chosen so that they would represent the variability in the data. While the testing dataset is small, it should still give a sufficient estimation of the performance of each neural network model. Accuracy is the proportion of correctly labeled pixels, measured separately for the validation and the testing dataset, both of which consist of unseen images for the CNNs. Intersection over unit (IoU) [8], also known as the Jaccard index, measures the average intersection over unit score for all classes. IoU considers both the false alarms and the missed values, as it is the ratio of correct pixels to the ground truth and predicted pixels in each class. BF-score [8] measures the F1 matching score of the boundaries as a value between 0 and 1, where larger value is better. BF-score is well suited for semantic segmentation as the contour quality is an indicator for the overall smoothness and quality of the segmentation. In addition to numerical evaluation, a comprehensive visual inspection was done on a larger number of images than that of the testing dataset.

In terms of accuracy and IoU, the modified U-Net models performed better on both the validation and the testing data compared to the DeepLabV3+ and SegNet networks. On all the networks, accuracy on test dataset is lower than on validation dataset, which is due to the testing dataset having more difficult images. Perhaps surprisingly, DeepLabV3+ model with Xception encoder achieved the best results on the BF-score measure. This is enforced by visual inspection, which showed that the DeepLabV3+ network outputs smooth looking lines. However, the output segmentations on the U-Net models looked more detailed, consistent, and better overall than on the DeepLabV3+ networks which struggled noticeably with unusually shaped, thin or vague annual rings and output blob-like segmentations on those difficult parts. SegNet was inferior to the U-Net and DeepLabV3+ variants on both the numerical and the visual validation.

Figure 29 shows example output of the CNN models implemented in Section 3.2 on an unseen image. The input image of Figure 29 is good for comparison as it features both thick and thin lines, saw cuts and a crack in the middle. Furthermore, there are annual rings that are clearly visible as well as some that are vague and partly broken. On SegNet, the overall quality of the segmentation is lacking compared to other networks. The resulting binary image contains noise and many false detections and SegNet also mistakenly detected saw cuts as annual rings, which the other neural network models did not. All the network models handled the crack well and did not mistake it for an annual ring. DeepLabV3+ variations handled thick and clear lines well, however, struggled significantly with small and unclear lines, which resulted in

		Accuracy (validation)	Accuracy (test)	MeanIOU	BF-score
U-Net	Dense blocks & VGG-19 encoder	0.920	0.841	0.704	0.976
	Dropouts & VGG-19 encoder	0.915	0.846	0.701	0.976
DeepLabV3+	Xception encoder	0.851	0.820	0.678	0.984
	MobileNetV2 encoder	0.845	0.811	0.660	0.977
SegNet	VGG-19 encoder	0.821	0.802	0.638	0.935

Table 2: Evaluation of the CNN models implemented in Section 3.2.

blob-like segmentations. U-Net models performed best overall and showed promising results. Similar characteristics of the neural network models as those described previously were visible on other test images as well.

DeepLabV3+ [7] had acquired better performance to that of U-Net and SegNet on many popular datasets such as Cityscapes and PASCAL VOC 2012. However, performance on other datasets is not a direct indicator of performance on the particular wood end-face dataset of this thesis, and as it turns out, U-Net achieved the best results of the networks tested in this chapter. This is likely due to skip connections in U-Net, which are capable of acquiring finer detail and give it an advantage over other networks in this specific task, which requires accurate segmentation of small details. The fact that U-Net had previously generalized well on similar applications such as medical image segmentation [27, 36], retinal blood vessel segmentation [25, 10] and satellite image segmentation [28] also indicated that it could be a suitable model for this problem.

Based on these results, the U-Net model with VGG-19 encoder and added dropout layers was chosen for the final implementation of this thesis and as a baseline network for moving forward and implementing a neural network for a real-time production version at the sawmill. The U-Net variant with dense blocks and VGG-19 encoder was similar in performance, however, slightly more complex and slower, therefore not chosen.

Section **??** further discusses the results of the annual ring detection task with a focus on possible future improvements to the current baseline network that is the U-Net based model, which obtained best results as presented in this section, as well as compares the results of deep learning approach versus image processing approach as a recap of this chapter.

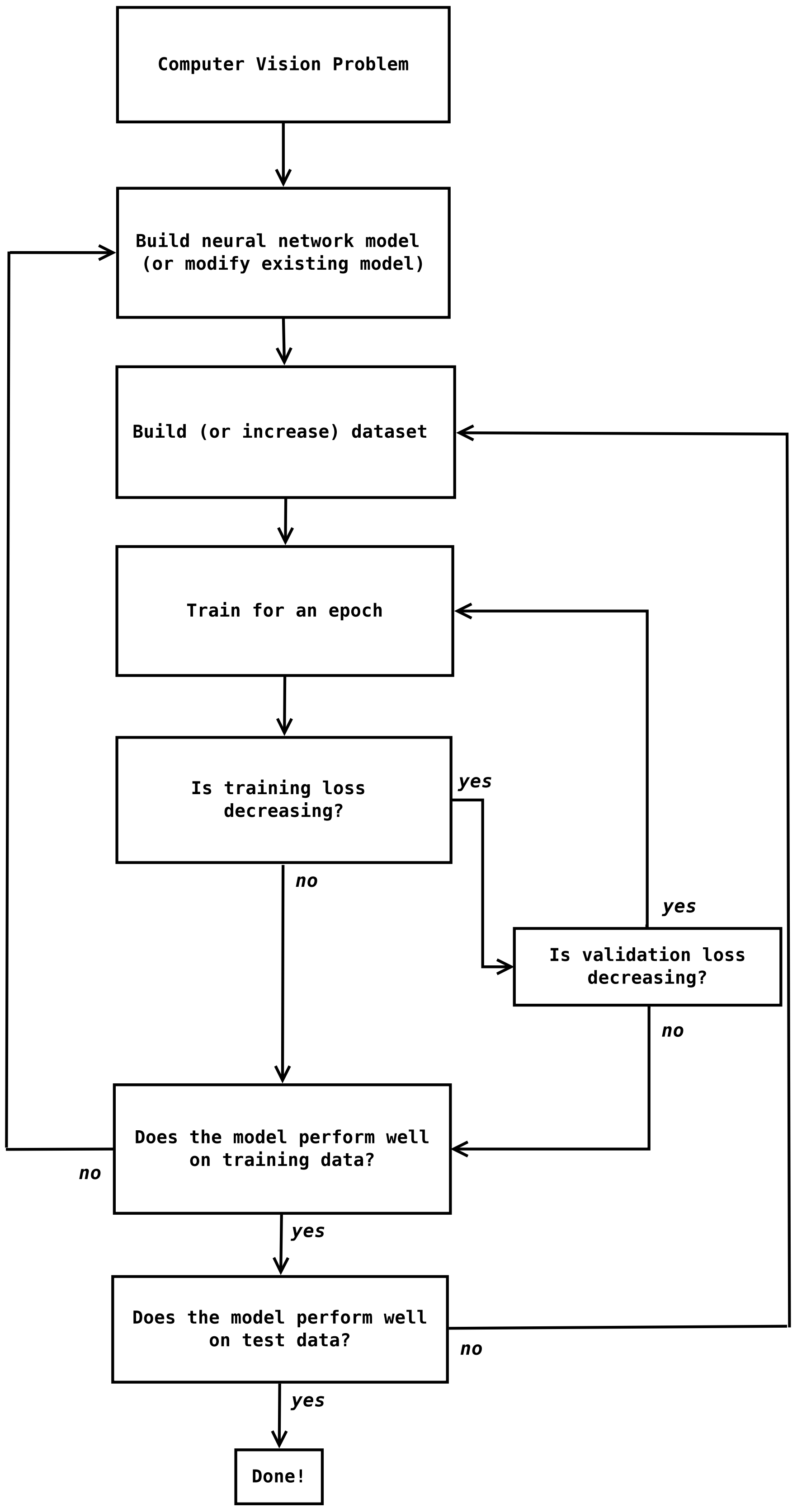
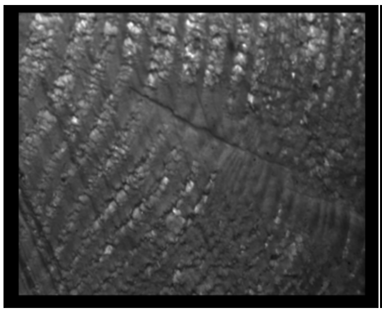
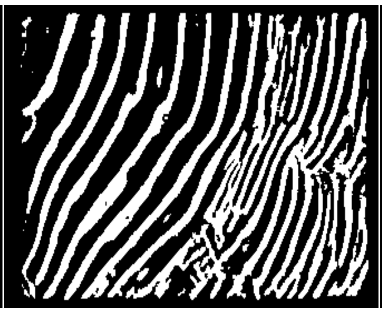
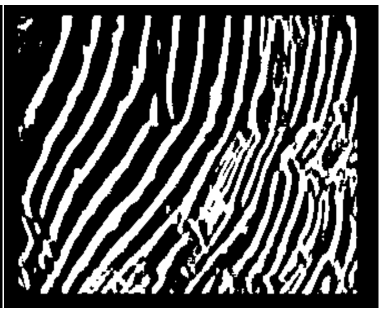
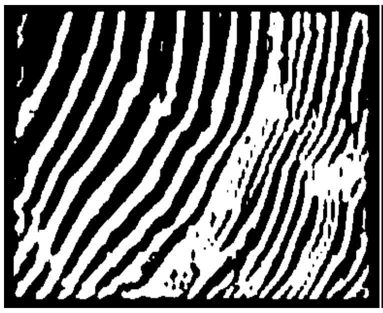

Computer Vision Problem
Build neural network model (or modify existing model)
Build (or increase) dataset
Train for an epoch
Is training loss decreasing?
yes
no
Is validation loss decreasing?
yes
no
Does the model perform well on training data?
no
yes
Does the model perform well on test data?
no
yes
Done!

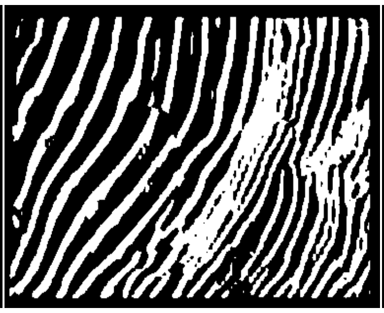
Figure 28: Workflow of training a deep learning model.


(a) Input image


(b) U-Net (dense blocks)


(c) U-Net (dropouts)


(d) DeepLabV3+ (MobileNetV2 encoder)


(e) DeepLabV3+ (Xception encoder)

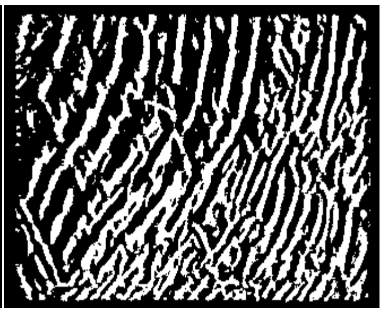

(f) SegNet

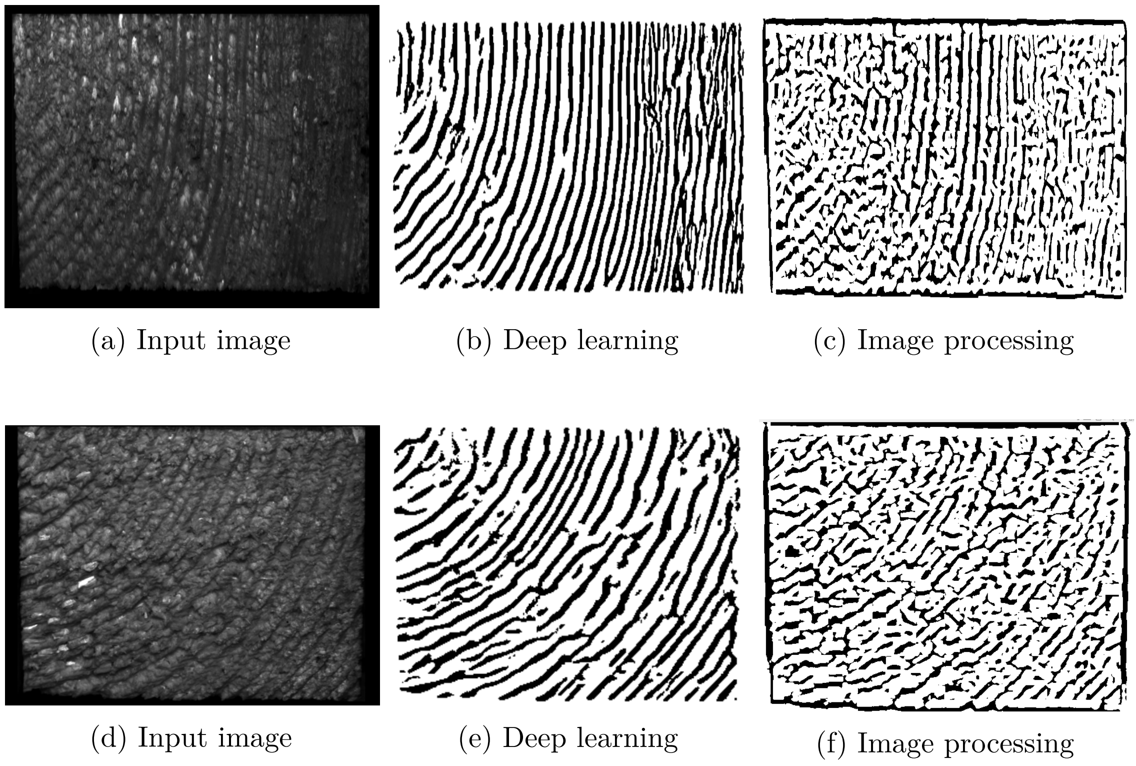
Figure 29: Examples of annual ring detections by different neural networks.

3.3 Results

The results of this chapter indicate the superiority of deep learning for this particular task. The annual rings do not have a specific intensity value, do not share a common shape, and are not perfectly circular or elliptical. In addition, there are various defects and noise caused by the rough and uncleaned condition of the wood end-faces. These facts make it very difficult, if not impossible, to develop a general and robust image processing based system for detecting annual rings on wood end-faces of varying conditions and properties. Indeed after trying various image processing methods and techniques in Section 3.1, the results were not desirable. Detecting annual rings by human eye and differentiating them from defects is effortless, which suggests that a convolutional neural network could similarly learn to detect annual rings on the wood faces. This was found to be true in Section 3.2, where a convolutional neural network was designed and trained for this task. Even with a relatively small amount of training data, the neural network was able to learn the general features of annual rings and was capable of differentiating between the annual rings and various defects, even on difficult condition wood end-faces. Furthermore, the convolutional neural network presented in Section 3.2 still has room for improvement. Another advantage of the deep learning approach over traditional methods is that it is robust to changes in the factory; the neural network would perform equally well if for example the lighting conditions or the camera position were to change, whereas a manually crafted system based on traditional image processing methods would have to be re-calibrated. A comparison of annual ring detection on a couple of images between the image processing and the deep learning method is shown in Figure 30.

More examples of annual ring detection on multiple images using the U-Net based network are shown in Figure 31. The resulting binary images look promising and prove that the network has learned to generalize on the data. Minor flaws in some segmentations should be fixable by improving the performance of the model through architectural modifications, larger and more comprehensive dataset, and better postprocessing. The network is a decent baseline model when moving forward towards actual production implementation in the sawmill. For a proof of concept version, it is successful and proves that the task of annual ring detection is doable even for the uncleaned and rough wood end-faces in a sawmill environment.

Having skip connections in the model appears to be necessary for this particular task where it is crucial to obtain fine details as the annual rings are thin and faint, based on the findings of trying different models, as discussed in Section 3.2.5. This suggests that a U-shaped model with skip connections is a good choice for this task. The field of semantic segmentation moves fast, and in literature, there are various re-imagined versions and modifications to the original U-Net model that have proven to outperform it in real-life applications. An example of these is U-Net++ [36], which features nested and dense skip connections that should better obtain fine details. The encoder and decoder parts in U-Net++ are connected through nested dense convolutional blocks in the skip pathways, which bridges the gap between encoder and decoder blocks and improves gradient flow. On four different medical image datasets, these improvements proved a clear advantage over the vanilla U-Net.



(a) Input image (b) Deep learning (c) Image processing

(d) Input image (e) Deep learning (f) Image processing

Figure 30: Comparison of annual ring detection results between the deep learning implementation of Section 3.2 and the image processing implementation of Section 3.1.

MultiResUNet [18] is another re-imagined version of U-Net, which, in particular, improved the performance on difficult images by introducing multi-residual blocks in place of the regular convolutional blocks. This modification allowed better feature flow between the encoder and the decoder parts. As discussed in the literature review of Section 2.1, Peng et al. [25] also modified the original U-Net architecture and achieved improved performance in a retinal blood vessel segmentation application, which resembles the annual ring detection task. Their CNN architecture, named CDNet, has a symmetrical shape as U-Net, however, uses segmentation blocks with short propagation paths which allows recovering more information from the propagation step. These and other features of the latest CNN architectures could be tested when moving forward in order to improve the model. Additionally, as discussed in Section 3.2.3, implementing more advanced and modern loss function variants in the pixel classification layer of the neural network could potentially improve the resulting segmentation maps.

Enlarging the dataset is the most obvious source for gaining improvement, as the initial dataset of 100 non-augmented images is quite small and could have more variance in the data to generalize better. The network performed especially well on wood end-faces that had clear and thick annual rings, which is likely due to laziness in building the dataset. It was easier to annotate images with thick annual rings as opposed to those wood faces that had many small and thin annual rings. Therefore when moving forward, these thin annual ring wood faces should be prioritized when choosing and annotating more training images.

Furthermore, as discussed in Section 3.2.4, trying more advanced postprocessing methods could improve the output of the segmentation by introducing more fine-grained detail, fixing broken annual rings, and removing noise.

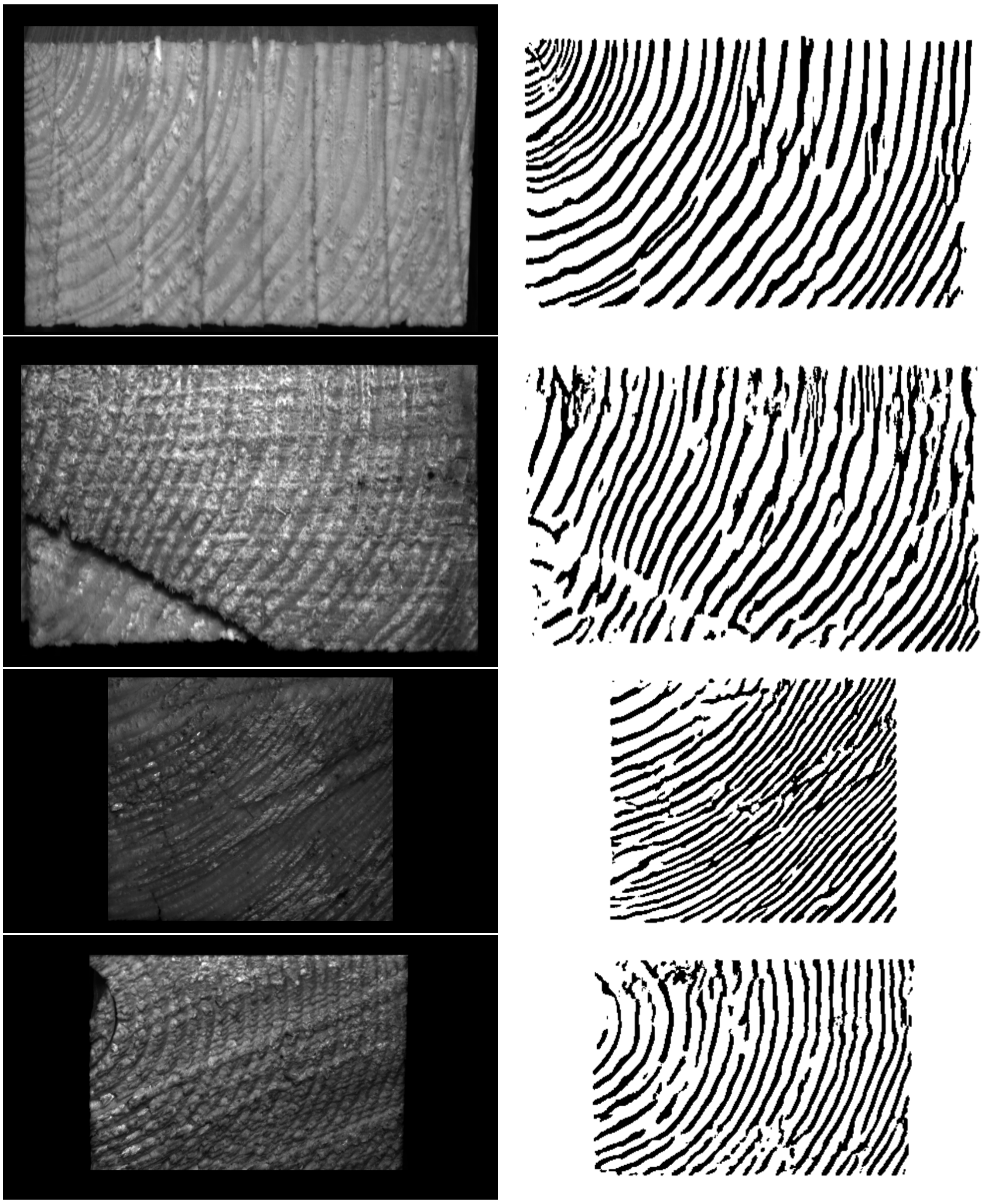


Figure 31: Example annual ring detections on unseen images using the U-Net based neural network model of Section 3.2.

4 Finding the Pith

Center (or the pith) of the wood end-face is found in order to calculate distances of the annual rings outwards from the pith. The pith is, however, rarely visible in the photos, as the wood is cut into blocks, so it should be searched outside of the image as well. This also makes it slightly more challenging to verify the results, as exact pith position cannot be accurately verified. The annual rings in the images are not perfectly circular, not identically shaped among themselves, and often too faint and partly broken, making it practically impossible to fit circles or ellipses into the image to find the center point. Annual rings should first be detected with the methods of Section 3.2, then, by getting the orientation and direction information of the rings, it becomes a mathematical problem to determine the pith position.

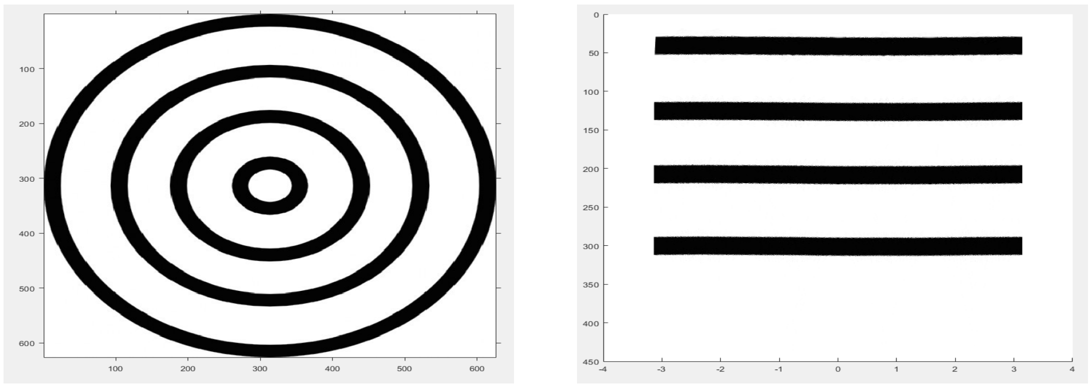
As reviewed in Section 2.1, multiple studies have been done on determining pith position of wood faces, but the majority of them were done for dendrochronology applications with clean and good quality wood and a clear circular pattern of annual rings, therefore not applicable to this thesis. Andreu and Rinnhofer [1] study was done on a sawmill setting, however they used a computer tomography scanner instead of a camera, and their method also requires a complete circular annual ring pattern and the pith itself to be visible. As far as the author is aware, there have not been other studies of pith estimation in sawmill setting on images taken with a regular camera except for Norell and Borgefors's paper [23] that could be viable in this case. Their method was discussed in detail in Section 2.1. Its data resembles that of this thesis, however the wood is not yet cut to pieces and the pith is visible in all the images. It was decided to use a new approach for this thesis instead.

The method was designed for this thesis by exploiting the (loosely) circular shape of annual rings. When transforming an image of a circle to the polar coordinate system at its center point, its circular lines become straight lines in the resulting polar image, as depicted in Figure 32. Similarly, the annual rings become approximately straight when transforming the wood end-face image to polar coordinates at the center point. This is visualized in Figure 33. It is a simple method, and it does not require the pith to be visible in the image. Additionally, the fact that circular rings can be viewed as straight lines is handy when calculating the ring distances, as will be explained in Chapter 5.

The polar coordinate system is a coordinate system where each two-dimensional point is determined by distance an angle from a reference point and direction. Cartesian coordinates x and y are converted to their corresponding polar coordinates r and ϕ using trigonometric functions:

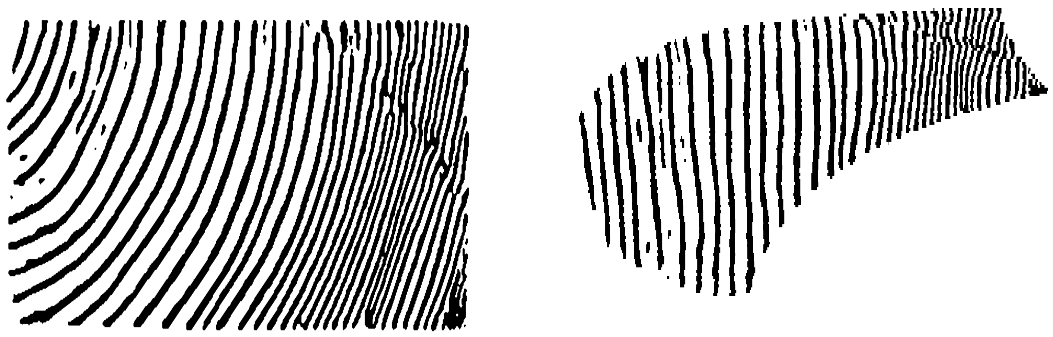
$$\begin{aligned} r &= \sqrt{x^2 + y^2} \\ \phi &= \arctan\left(\frac{y}{x}\right) \end{aligned} \tag{13}$$

Histogram of Oriented Gradients (HoG) [9] was chosen as the method for determining the straightness of lines in a polar transformed image due to it is simplicity and computational efficiency. Alternative methods for solving this task could be based



(a) Image in Cartesian coordinates (b) Image in Polar coordinates

Figure 32: An image of concentric circles transformed to polar coordinates.



(a) Image in Cartesian coordinates (b) Image in Polar coordinates

Figure 33: An image with annual rings detected transformed to polar coordinates at approximate center point.

on, for example, image moments. HoG is essentially a feature detector, where the distribution of gradient intensities determines the local shape and appearance of the object. The image is first divided into blocks, then gradient magnitude and direction at each pixel is calculated using a Sobel operator (see Figure 11 in Section 3.1.2) with a kernel size of 1. Using a block size of 25 by 25 was found to best capture the desired features in this case. The annual rings do not fill the whole image, and few of the blocks are usually empty, those blocks are left out of the calculations. Furthermore, using unsigned gradients, i.e., converting orientations from 0°–360° to that of 0°–180°, performed better in Dalal and Trigg's study of human detection [9] and also in this particular problem of finding the straightness of lines. These gradient magnitude and orientation values at each pixel then act as weighted votes when assigning values to histogram bins. Nine histogram bins spreading evenly over the 180° spectrum were created, as visualized on Figure 34. The weighting of the votes was chosen to be based on the magnitude and the distance to the nearest bins: the votes contribute proportionally to the nearest bins. For example, a pixel with 70° and magnitude of 14 would be halfway between the 60° and the 80° bins, thus a value of 7 is put to both the 60° and the 80° bins. Similarly, if a pixel would have an orientation of 20° and a magnitude of 10, a value of 10 would be put to the 20° bin. An extra detail to be aware of is that when the angle is greater than 160°, it wraps around and contributes proportionally to the 0° and the 160° bins. After building the histogram bins, the HoG method would then proceed by calculating a HoG feature vector, but in this case, the histogram bins can be used to formulate an optimization problem for maximizing the straightness of the lines, which is the point where the location of the pith is.

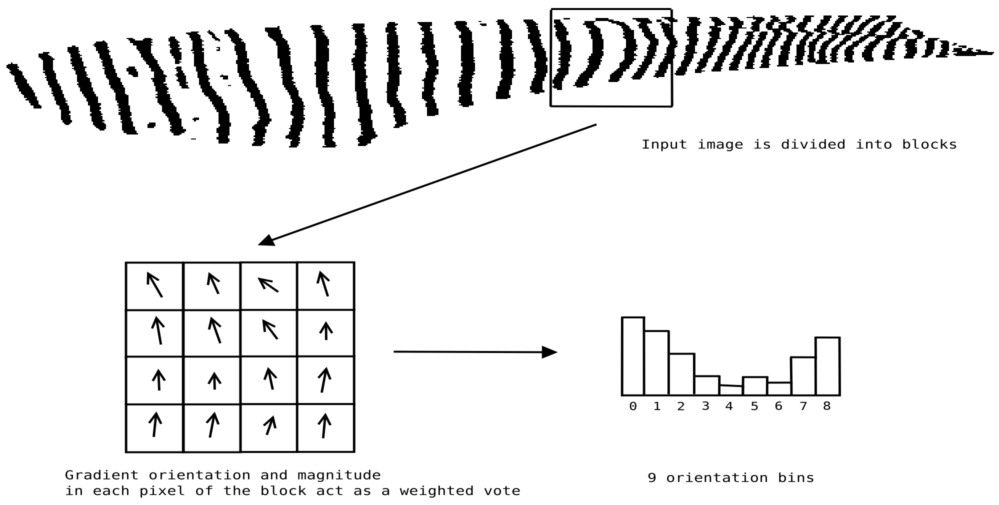

Input image is divided into blocks
0 1 2 3 4 5 6 7 8
Gradient orientation and magnitude
in each pixel of the block act as a weighted vote
9 orientation bins

Figure 34: Visualization of the Histogram of Oriented Gradients method.

Section 4.1 further explains the algorithm and presents a pseudocode solution.

4.1 The Algorithm

The pith detection algorithm operates by transforming the image to polar coordinates at several points of the binary image and then builds a Histogram of Oriented Gradients for determining the vertical straightness of the resulting lines in small patches of the image. The more straight the lines in the vertical direction, the closer the point is to the pith. The score for describing the straightness of lines in a block are calculated for each block by summing the gradient bins of HoG. The bins are multiplied with bias variables that can be controlled in order to improve the results. The calculation of the gradient bins was described in Section 4. The bias variable for the bins corresponding to 0° and 160° should be the highest as they correspond to closely straight lines. Other near straight bins should get a lower value, and those corresponding to horizontal angles should get a zero or negative bias value. After the score for each block has been calculated, an average of all blocks is calculated in order to get a single value describing the straightness of the lines. The simplest way to find the pith with this method would be by brute force: transform the image to polar coordinates at all points and find the point which maximizes the vertical straightness value. That would, however, be inefficient and too slow, considering the running time requirement (one second) of this system. The search space is instead narrowed down by first transforming the image to polar coordinates at each corner and middle point of the image and finding the point with the maximum straightness. This way, an approximate location of the pith is found, and the correct center point can then be searched for near this approximate location, in a much smaller search space than with a brute force approach.

By observing that the calculation of the straightness score for polar transformations at different points of the image is not dependent of each other, it can be noticed that parallelism [32] can be utilized in order to do simultaneous calculations of the straightness scores, which improves the speed of the process substantially. Similarly, the calculation of HoG for each block can be done independently. On a four-core processor, this could potentially result in a speedup by a factor of 4, as by running the calculations independently on four threads, the operating system will likely assign these four threads to each of the four CPU cores. There are different ways of implementing thread creation in the code, depending on the operating system and the programming language. Instruction level parallelism happens in the CPU automatically if the calculations are organized so that there is potential for parallelism; in other words, the CPU assigns instructions to independent threads by itself. This can be exploited by merely organizing the calculations on independent variables. Furthermore, one register in a modern CPU can store 8 floats in its 256-bit register and manipulation of vectors is highly optimized and efficient, therefore in addition to instruction-level parallelism, the algorithm can be further speed up by vectorization which allows more parallel operations. In order to allow vector instructions, the compiler has to be instructed to do so by using vector types that are arrays of 8 floats and by again ensuring that the code is organized to have independent operations. Proper memory alignment has to be ensured if using pointers to the vector type or the algorithm may crash.

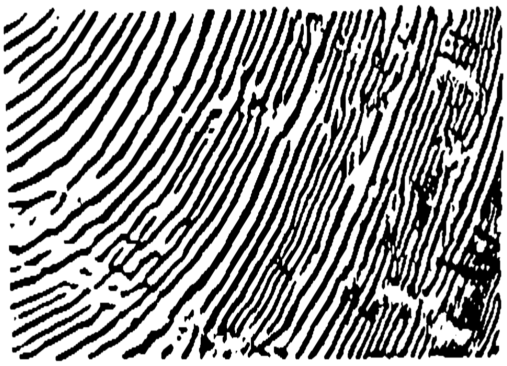
With parallelization, it was possible to reduce the running time of the pith detection algorithm from over a second to a fraction of second (≈ 0.15 s). Further improvements to the speed could be done by utilizing more advanced techniques [32] such as prefetching or through better use of cache memory, but the running time obtained with previously described parallelization techniques was sufficient for this project. The total running time of the whole system is discussed in Chapter 6. Additionally, using GPU instead of CPU in most computers including that of this thesis enables vastly more operations per second but at lower clock speed thus requires careful parallelization of the code in order to gain improvement; it would be more difficult to implement and not worth the time and effort at least in a prototype level implementation.

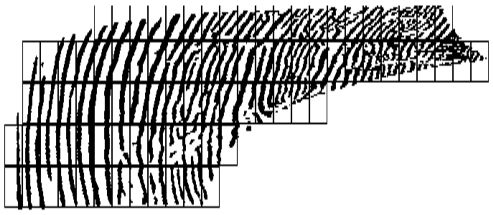
Algorithm 1 Pith Detection

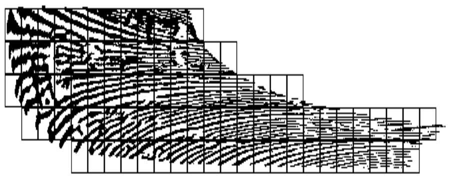
```
procedure FindPith(img)
    Vector<Point(x,y)>   startingPoints[8]
    Vector<float>   Scores[8]
    for i  ←  0  to  Size(Scores) do                              ▷ Algorithm 2
        ScoreAtXY(img, startingPoints[i].x,startingPoints[i].y, 4,1,0)
    ind ← distance(begin(Scores), max(Scores))
    iterPoint ← startingPoints[ind]
    Vector<Point(x,y)>   newPoints[8]
    Area ← 200
    Skip ← 50
    for i  ←  -Area/2  to  Area do
        for j  ←  -Area/2  to  Area/2 do
            newPoints.pushback(Point(iterPoint.x + i, iterPoint.y + j))
            i ← i + Skip
            j ← j + Skip
    Vector<float>   newScores[size(newPoints)]
    for i  ←  0  to  Size(newScores) do                           ▷ Algorithm 2
        ScoreAtXY(img, startingPoints[i].x,startingPoints[i].y, 5,1,0)
    bestInd ← distance(begin(newScores), max(newScores))
    return newPoints[bestInd]
```

Algorithm 2 Straightness score

```
procedure ScoresAtXY(img, x, y, bias1, bias2, bias3)
    polarimg ← WarpPolar(img,img, Point(x,y))
    gx ← Sobel(polarimg, dx=1, dy=0, ksize=1)
    gy ← Sobel(polarimg, dx=0, dy=1,0, ksize=1)
    angle, mag ← Cart2Polar(gx,gy)
    N ← 25
    thres ← 0.1*(N*N)
    counter, score ← 0
    for y ← 0 to height(img) - N do
        for x ← 0 to width(img) - N do
            block ← rect(x,y,N,N)
            if numWhitePixels(block) $\leq$ thres then
                continue
            mags[9] ← buildHoG(block, angle, mag)
            sumMags ← $\sum_{n=0}^{8}$ mags[$n$]
            for n ← 0 to 8 do
                mags[n] ← mags[n] / sumMags
            score ← score + (bias1 * (mags[0] + mags[8])
            + bias2 * (mags[1] + mags[7])
            + bias3 * (mags[2] + mags[3] + mags[4] + mags[5] + mags[6])
            counter ← counter + 1
            y ← y + N
            x ← x + N
    return score/counter
```


(a) Input image


(b) Top left corner ($x = 25, y = 25$) straightness score: 1.374


(c) Top right corner ($x = 975, y = 25$) straightness score: 0.382

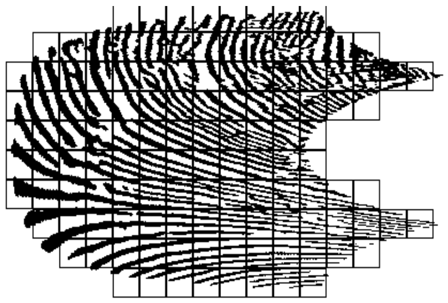

(d) Top middle ($x = 500, y = 25$) straightness score: 0.803

Figure 35: Example of running the straightness score algorithm on different points of an input image (real center is near the top left corner).

4.2 Results

Due to the pith not being visible in most of the pictures, accurate ground truth positions cannot be established to calculate the precise offset of the estimated pith positions. A visual inspection and evaluation is therefore used to verify the results of the algorithm.

Pith estimation relies on the success of the tree ring segmentation of Chapter 4. When the segmentation is successful and nearly flawless, the algorithm estimates the pith position well in the vast majority of cases. However, if the segmentation contains false information, then the pith estimation is usually also slightly off. However, the pith estimation does not have to be perfect for the ring detection to succeed, as will be described in the next chapter. Figure 36 shows a collage of pith detections by the algorithm presented in this chapter. As can be seen, the algorithm finds the approximate location on wood faces of different sizes with varying annual ring patterns. These images in Figure 36 had a fairly successful segmentation however not completely flawless.

The pith estimation is sufficiently good on approximately 90.4 % of the cases based on manual inspection of 80 test images. False detections are most often caused by flaws in the annual ring detection, therefore the pith detection algorithm should be thoroughly tested again in the production version once the neural network has been improved. In rare cases, weirdly shaped annual rings that do not resemble circular or elliptical shapes fail the pith detection and cause poor results. Given the subset of images that the algorithm was tested on, there were four of these cases. The share of the weirdly shaped rings should be determined through investigating a larger subset of wood end-face images, such as 1000 images, to get a more accurate estimate. If the share of wood end-faces where this algorithm fails to detect the pith is significant, further modifications should be made in future work, such as by detecting the approximate circularity of the rings and then developing a different method for finding the pith on these non-circular shaped rings.

On an Intel Core i7-8550U (8×4.0 GHz) processor, the average running time out of 50 iterations was 0.148 seconds. The capacity of the CPU computational power was better utilized through parallelization and vectorization, which considerably increased the speed of the pith detection algorithm, as explained in Section 4.1. The total running time of this system is discussed in the concluding Chapter 6.

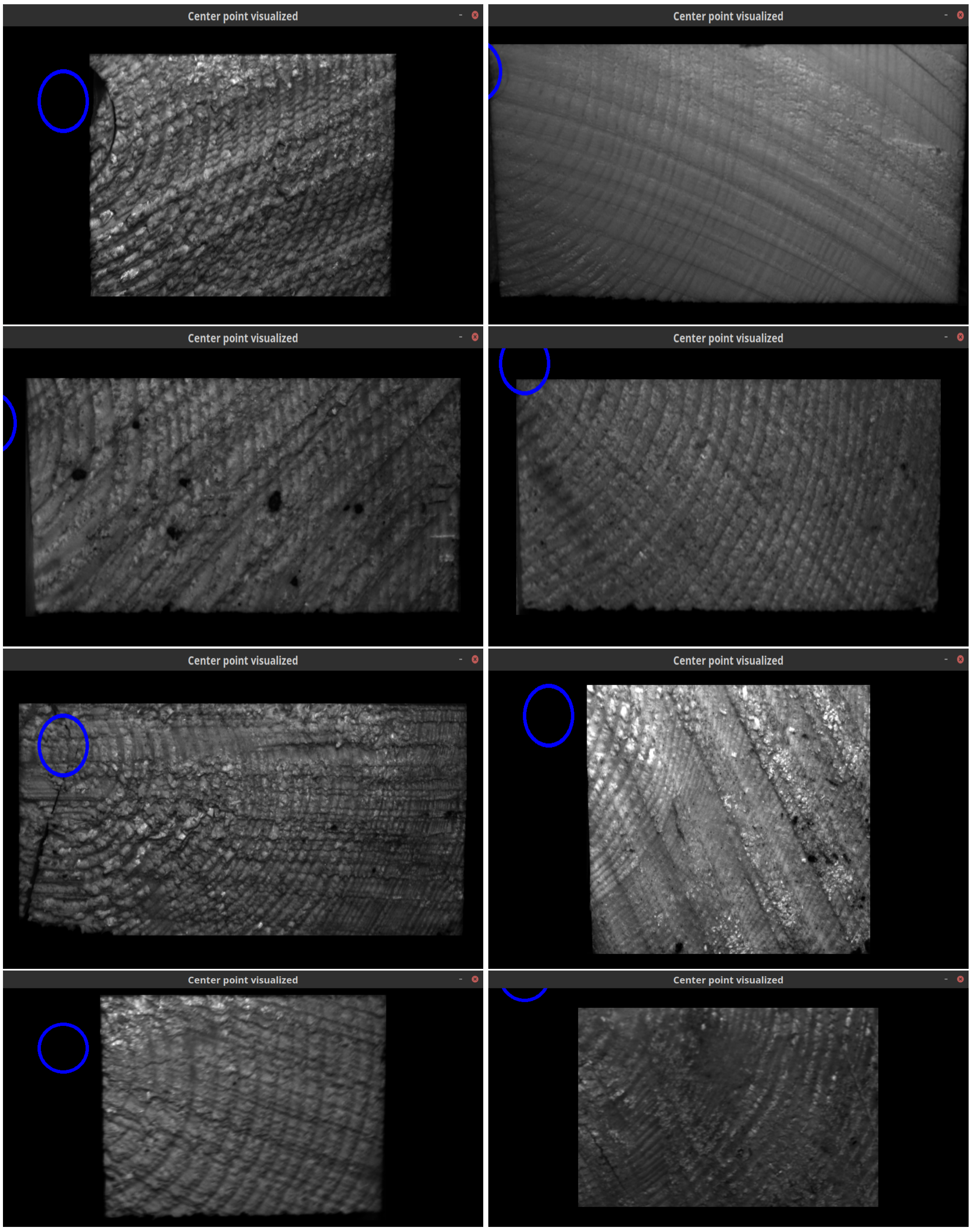


Figure 36: Example pith detections.

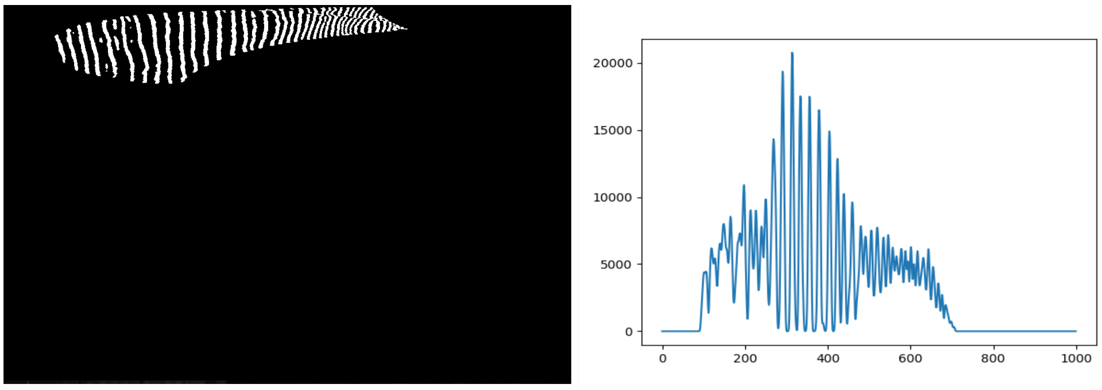
5 Calculating Ring Distances

The final step of the system is to calculate the annual ring distances. It is a straightforward task as it is, but there are still few aspects to take care of: pixel distances should be precisely converted to millimeter distances, noise should be filtered, sanity checking and corresponding actions should be made.

This chapter is organized as follows: Section 5.1 presents a method for calculating ring distances, followed by Section 5.2, which discusses unit conversion from pixels to millimeters and Section 5.3 evaluates the results of annual ring distance calculation.

5.1 Computation

Annual ring distance calculation algorithm designed for this thesis uses polar transformation with the center point estimate presented in the previous chapter. As stated in the center finding chapter, transforming the wood end-face image into polar coordinates at the approximate center point results in the annual rings to appear as approximately straight lines in the resulting polar image. This makes annual ring distance calculation straightforward: by calculating horizontal projection (or horizontal profile), that is, in this case, a histogram of white pixels in the horizontal axis, we get information of the annual ring locations as peaks in the x–axis. Then it is possible to calculate distances between these peak locations minus the width of the rings in order to get the distances between the annual rings. More specifically, the projection is a vector of size equal to image width, so that each index represents a horizontal location and each value equals the number of white pixels in that location. An alternative to taking a horizontal projection that considers all white pixels in the image could be, for example, to draw multiple thin histogram lines from left to the right and calculate the average of ring distances from multiple histograms.



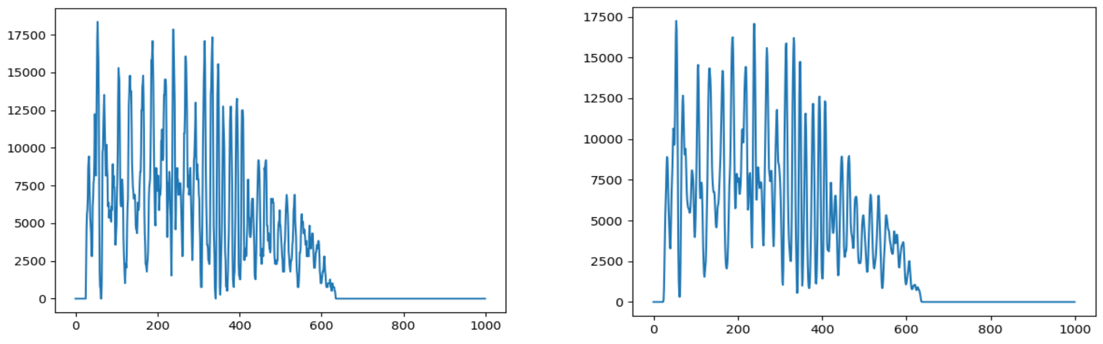
(a) Input image (b) Horizontal projection histogram

Figure 37: Binary image of annual rings transformed to polar coordinates at its pith and its corresponding horizontal projection.

Likelihood of error detections should also be taken into account, as not all the white pixels necessarily belong to annual rings given the minor flaws in annual ring segmentation of Chapter 4, therefore a small and vague peak may not describe the horizontal location of an annual ring. A small amount of noise and error detections should not be a severe problem however, as it can be filtered out by smoothing the histogram and by ignoring non-prominent peaks. Noise removal in the histogram was done by applying Gaussian smoothing, which is a convolution between the input image (histogram) and a Gaussian function, defined mathematically as [14]:

$$\begin{aligned} h(x,y) &= e^{-\frac{x^2+y^2}{2\sigma^2}} \\ g(x,y) &= f(x,y) * h(x,y), \end{aligned} \tag{14}$$

where $*$ is a convolution operation, σ is the standard deviation for controlling the shape of the Gaussian filter, $f(x,y)$ is the original histogram and $g(x,y)$ is the filtered histogram. The result of applying Gaussian smoothing of size 7×7 with $\sigma = 1$ on the histogram is depicted in Figure 38 below.

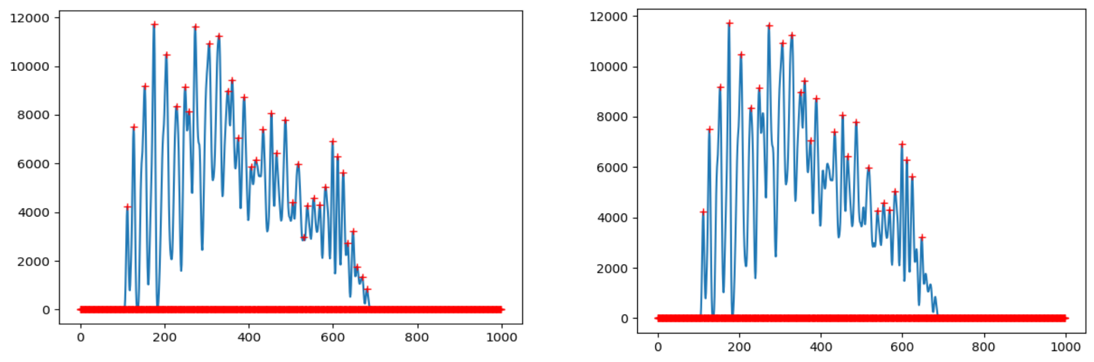


(a) Input histogram (b) Smoothed with Gaussian blur

Figure 38: Horizontal projection histogram before and after smoothing.

The peaks are the local maxima of the horizontal projection vector. However, even after applying the Gaussian smoothing, there are still some negligible peaks which do not represent an annual ring but are instead noise introduced in the annual ring detection. Prominent peaks are clear peaks that confidently belong to annual rings and should be found. One way of finding these desired peaks is to analyze their persistence, which is the difference between the value of a local maximum and its corresponding local minimum. For classifying a peak as prominent or not, it was found that calculating the persistence of all peaks and then multiplying the average persistence with an experimentally chosen threshold coefficient value was successful at filtering vague peaks from the horizontal projection histogram. A threshold coefficient value of 0.5 was found to be slightly too aggressive, it keeps only the most prominent peaks and may ignore some annual rings, whereas a value of 0.15 did not filter out all erroneous peaks. The value of 0.3 was determined to have a right balance in ignoring non-prominent peaks and keeping clear peaks. Figure 39 shows the result of finding all peaks and the result of finding only the prominent peaks by filtering out peaks that are smaller than $0.3 \cdot m$, where m is the mean persistence.

After finding and filtering the peaks from the histogram, they should now correspond to the locations of the annual rings. This is a place for sanity checking to take



(a) All peaks (b) Filtered peaks

Figure 39: Found peaks (marked as red cross) visualized on the horizontal projection histogram.

place: while the distance between annual ring locations is not constant and can vary a lot, they should still be in a somewhat similar range. If there is an absurdly long distance between some peaks in comparison to other distances, say a 75 pixel distance between two peaks whereas other peaks would have a distance of 5 to 25 pixels, then there is likely an error. This could be due to a defect in the wood end-face such as a knot or thick layer of tar. An erroneous distance would ruin the calculation of average ring distance, therefore it should be removed from the array of annual ring distances. Upon testing different methods for finding and removing a false distance, a threshold value was determined to be defined simply as $2 \cdot \text{mean}(distances)$, where *distances* is the array holding all the found distances. This value was found to be successful in removing false detections while preserving correct detections. This should, however, be more thoroughly tested in order to see if it is a reliable threshold value but it is sufficient for a prototype implementation.

5.2 Unit Conversion

The final step before deciding whether or not the woodblock is too fast grown or normally grown is to convert pixel distances to millimeters. It is quite a small task, however it is highly important to do the conversion accurately. This pixel to millimeter conversion was not implemented at the time of writing this thesis, however it should be done next when moving forward from the PoC phase of the project. While the unit conversion was not tested in practice, the following paragraph discusses the problem in theory.

Norell presented a couple of ways to determine pixel size in her thesis [22]. First of those methods was using a picture of a reference object at the same distance from the camera as the wood end face, assuming there is no perspective distortion in the image. In the second method, the wood faces were images slightly from above, which causes perspective distortion. Information about the distance between the camera and the wood face, the focal length of the camera, and rotation of the object were used. After detecting the locations of the calibration points with edge detection and line fitting, the obtained real-world coordinates were transformed to camera coordinates by a 3×3 rotation matrix followed by a translation along the coordinate axes. The three-dimensional camera coordinates were then transformed into two-dimensional image coordinates by knowing the focal length and distance to the object. This second method is somewhat similar to that of this thesis, as the wood faces are likewise imaged from above, as shown in Figure 3 of Section 2.2. Before implementing the unit conversion, the focal length and distance should therefore be determined and the conversion calibrated according to some reference object. A more simple way than using an external reference object would be to use a wood end-face itself as a reference, as the sawmill workers know the dimensions of ongoing wood planks. Lens distortion can be considered insignificant as the wood end-faces are located in the middle of the viewfinder, while lens distortion tends to occur in the corners.

5.3 Results

Given a sufficiently well succeeded annual ring detection and pith location estimation, the annual ring distance calculation should robustly calculate the annual ring distances. A small amount of noise or erroneous pixels not belonging to annual rings are not a problem as these are filtered out from the histogram and only prominent peaks that confidently belong to annual rings are taken into account. Furthermore, missing annual rings due to a defect such as a knot are spotted by the algorithm and do not interfere with the calculation of mean distances. By calculating distances between annual rings in pixels manually from the images, it was possible to estimate the accuracy of the algorithm, which appears to match the manually measured distances when the annual ring detection and pith estimation had succeeded. Again, this should later be precisely and comprehensively tested by comparing to accurate ground truth, once the CNN has been improved and the pixel to millimeter conversion has been made, before the system can be trusted to operate live in the production line of the sawmill.

Performance-wise, this is a simple and fast linear time algorithm. The current implementation runs on a single core and could be further improved with similar parallelization techniques as described in Section 4.1. The running time is however only a fraction of that of the whole system, thus improving the overall performance of the system should be done by focusing on improving the bottleneck step of the system, which dictates the majority of the running time, the annual ring detection.

The following conclusion chapter evaluates the annual growth analysis system implemented in this thesis as a whole from both an academic and a practical perspective and discusses its impact, rate of success, and future work.

6 Conclusions

This thesis has developed software for automatic analysis of wood growth in a sawmill environment using deep learning methods. The proposed system provides a general solution for detecting annual rings and calculating their distances on wood end-faces of varying conditions, as well as performed considerably better than traditional image processing methods. For a prototype system, the results of this thesis are promising and demonstrate that deep learning is applicable to the task of annual ring detection and can reliably carry out automatic wood growth analysis in a sawmill.

The proposed software consists of four distinct steps: preprocessing, annual ring detection, pith estimation, and ring distance calculation.

Preprocessing detects and extracts the correct wood end-face from an input image acquired from the factory line using simple thresholding and binary image manipulation techniques. Due to an IR filter pointing towards the wood end-face, it appears brighter than the rest of the image, making the extraction task robust and straightforward.

Annual rings were detected using deep learning. In the recent years, deep learning has been widely used for many computer vision problems similar to that of annual ring detection with great success. Additionally, since detecting annual rings is an effortless task for human operators, a convolutional neural network could also be well suited for the task. This thesis also implemented an image processing based solution in order to compare the annual ring detection results to those of the neural network implementation. Even on a small training dataset, the proposed neural network performed substantially better than the image processing implementation and was capable of generalizing the features of the annual rings.

Future work could focus on further improving the neural network. Enlarging the training dataset would improve the performance of the model, as it was quite small, and allow training for a longer epoch. Although the network implemented this thesis performed well on wood end-faces with thick annual rings, it struggled with thin and unusually shaped annual rings. This is due to the training dataset having few examples of such difficult annual rings. Adding more variance (especially images with thin annual rings) to the dataset would allow the neural network to generalize better on different types and shapes of annual rings. In addition to the dataset, the neural network model could be modified in order to improve performance. Higher performance could be achieved by studying newer, more advanced network architectures. However, having a U-shaped model with skip connections enables U-Net to perform well on this particular annual ring detection task which requires accurate segmentation of fine detail despite lacking a large dataset. Therefore, a similar type of architecture should be used in future work as well. Newer U-shaped network architectures have outperformed the original U-Net model in medical image segmentation tasks by using methods derived from other state-of-the-art network models, such as denser convolution blocks in the skip connection pathways and multi residual convolutional blocks. Similar modifications could be used in this task as well. Additionally, using more advanced loss functions and postprocessing could also improve the accuracy of the resulting segmentation.

Finding the position of the pith was needed for calculating annual ring distances. In the data of this thesis, the pith remains most often outside the images, and the wood blocks were cut into pieces, thus preventing the use of circle detection techniques to detect the center point. The pith detection algorithm exploits the loosely circular shape of the annual rings by using polar transformation to estimate the pith position by detecting the point where the annual rings become as straight as possible. The algorithm was able to estimate the pith robustly when the annual ring detection step was successful; however, unusually shaped annual rings could sometimes lead to errors in the pith estimation. After finding the pith, annual ring distances were by taking a horizontal projection of the image at polar coordinates, and then determining the distance between the peaks in the histogram, thus enabling the algorithm to obtain the distances between the annual rings. For a prototype software, both the pith estimation and annual ring distance calculation were sufficient. However, they should be more thoroughly tested before relying on them in a real application running in the sawmill. Possible modifications could be made later based on the results of testing and validation.

The total running time of the system was slightly higher than the required 1 second. A clear bottleneck of the software is the annual ring detection part, which took over half a second. This issue could, however, be easily improved in future work by running the segmentation on a GPU, as this is currently being computed on a CPU. This change should reduce the total running time to less than 1 second.

References

[1] Andreu, J-P., and Rinnhofer, A. Enhancement of annual rings on industrial CT images of logs. *IEEE Proceedings of the 16th International Conference on Pattern Recognition (ICPR 2002)*, IEEE, Vol. 3, 2002, pp. 261–264.

[2] Arthur, D., and Vassilvitskii, S. k-means++: The Advantages of Careful Seeding. *SODA '07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, January 2007, pp. 1027–1035.

[3] Badrinarayanan, V., Kendall, A., and Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, Vol. 39, 2017, pp. 2481–2497.

[4] Borgefors, G. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, Vol. 34, 1986, pp. 344–371.

[5] Buduma, N. *Fundamentals of Deep Learning.* O'Reilly media, 2017.

[6] Cerda, M., Hitschfield-Kahler, N., and Mery, D. Robust Tree-Ring Detection. *Proceedings of the Second Pacific Rim Symposium on Advances in Image and Video Technology (PSIVT 2007)*, Springer, LNCS volume 4872, 2007, pp. 575–585.

[7] Chen, L-C., Zhu, Y., Papandreou, G., Schroff, F., and Hartwig, A. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. *European Conference on Computer Vision (ECCV (7) 2018)*, Springer, LNCS volume 11211, August 2018, pp. 833–851.

[8] Csurka, G., Larlus, D., and Perronnin, F. What is a good evaluation measure for semantic segmentation? *British Machine Vision Conference*, BMVA Press, 2013.

[9] Dalal, N., and Triggs, B. Histograms of Oriented Gradients for Human Detection. *International Conference on Computer Vision & Pattern Recognition (CVPR '05)*, IEEE, Vol 1, June 2005, pp. 886–893.

[10] Dasgupta, A., and Sing, S. A Fully Convolutional Neural Network Based Structured Prediction Approach Towards The Retinal Vessel Segmentation. *Proceedings of the 14th IEEE International Symposium on Biomedical Imaging (ISBI 2017)*, IEEE, 2017, pp. 248–251.

[11] DeepSystems, 2019, https://supervise.ly. Accessed 14 Nov 2019.

[12] Fabijanska, A., and Malgorzata, D. DeepDendro - A tree rings detector based on a deep convolutional neural network. *Computers and Electronics in Agriculture*, Vol 150, July 2018, pp. 353–363.

[13] Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., and Garcia-Rodriquez, J. A Review on Deep Learning Techniques Applied to Semantic Segmentation. *ArXiv:1704.06857*, April 2017.

[14] Gonzalez, R., C., and Woods, R. E. *Digital Image Processing (3rd edition).* Pearson Education International, 2008.

[15] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning.* MIT Press, 2016.

[16] Hong, L., Wan, Y., and Jain, A. Fingerprint Image Enhancement: Algorithm and Performance Evaluation. *Pattern Analysis and Machine Intelligence*, IEEE, Vol. 20, No. 8, August 1998, pp. 777–789.

[17] Huang, G., Liu, Z., and van der Maaten, L. Densely Connected Convolutional Networks. *Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017, pp. 2261–2269.

[18] Ibtehaz, N., and Sohel Rahman, M. MultiResUNet: Rethinking the U-Net Architecture for Multimodal Biomedical Image Segmentation. *Neural Networks*, Vol 121, pp. 74–87, February 2019.

[19] Long, J., Shelhamer, E., and Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2015, pp. 3431–3440.

[20] Molder, A., and Martens, O. Image Processing in the Woodworking Industry: Challenges, Solutions and Platforms. *Electronics and Electrical Engineering*, No. 7, Vol 113, 2011.

[21] Norell, K. Automatic Counting of Annual Rings on Pinus Sylvestris End Faces In Sawmill Industry. *Computers and Electronics in Agriculture*, no 2, Vol 75, 2011, pp. 231–237.

[22] Norell, K. Automatic Analysis of Log End Face Images in the Sawmill Industry. Doctoral Thesis, 2010.

[23] Norell, K., and Borgefors, G. Estimation of pith position in untreated log ends in sawmill industry. *Computers and Electronics in Agriculture*, No 2, Vol 63, 2008, pp. 155–167.

[24] Otsu, N. A Threshold Selection Method from Grayl-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, IEEE, No. 1, Vol. 9, 1979, pp. 62–66.

[25] Peng, S., Zheng, C., Xu, F., Xiao, H., and Nam, H-D. Blood Vessel Segmentation by Using CDNet. *2018 3rd IEEE International Conference on Image, Vision and Computing (ICIVC)*, IEEE, 2018, pp. 305–310.

[26] Qin, X., Zhang, Z., Dehghan, M., and Jagersand, M. ByLabel: A Boundary Based Semi-Automatic Image Annotation Tool. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2018, pp. 1804–1813.

[27] Ronneberger, O., Fischer, P., and Brox, T. U-Net: Convolutional Neural Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Springer, LCM 9351, May 2015, pp. 234–241.

[28] Sedov, A.G, Khryaschev, V.V, Larionov, R.V, and Ostrovskaya, A.A. Loss Function Selection in a Problem of Satellite Image Segmentation Using Convolutional Neural Network. *2019 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO)*, IEEE, 2019, pp. 1–4.

[29] Shorten, C., and Khosgoftaar, T. A Survey on Image Data Augmentation for Deep Learning. *Journal of Big Data 1/2019*, Vol 6, July 2019.

[30] Simonyan, K., and Zisserman, A. Very Deep Convolutional Networks For Large-Scale Image Recognition. *International Conference on Learning Representations (ICLR 2015)*, April 2015.

[31] Sudre, C., Vercauteren, T., Ourselin, S., and Cardoso, J. Generalised Dice Overlap as a Deep Learning Loss Function For Highly Unbalanced Segmentation. *Deep Learning Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, Springer, LCM 10553, 2017, pp. 240-248.

[32] Suomela, J. Programming Parallel Computers, 2019, https://ppc.cs.aalto.fi. Accessed 6 Nov 2019.

[33] Tomasi, C., and Manduchi, R. Bilateral Filtering for Gray and Color Images. *Proceedings of the 1998 IEEE International Conference on Computer Vision*, IEEE, 1998, pp. 839–846.

[34] Willis, A.J., and Myers, L. A cost-effective fingerprint recognition system for use with low-quality prints and damaged fingertips. *Pattern Recognition*, No 2, Vol. 34, 2001, pp. 255–270.

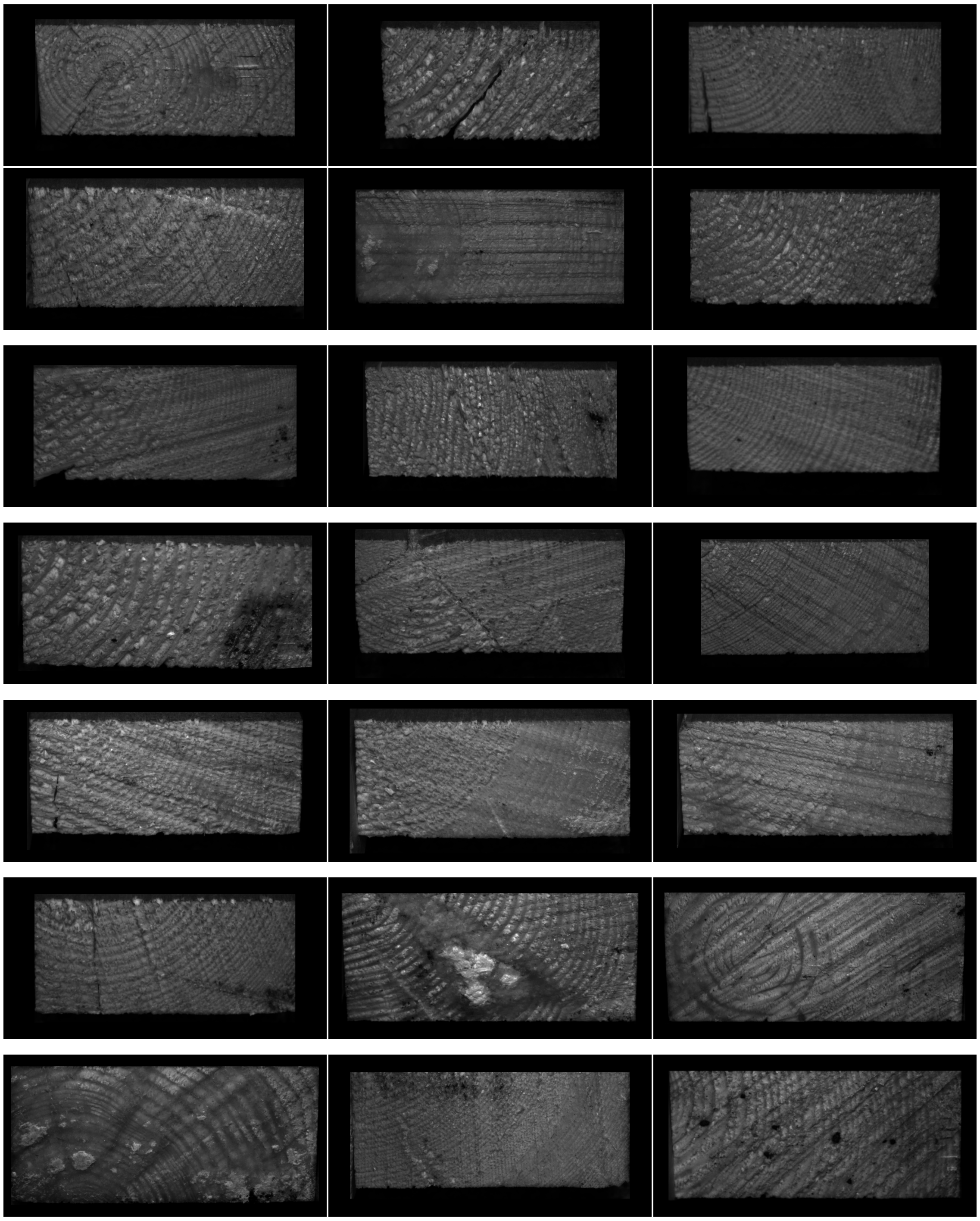
[35] Yu, C., Xie, M., and Qi, J. An Effective and Robust Fingerprint Enhancement Method. *2008 International Symposium on Computational Intelligence and Design*, IEEE, October 2008, pp. 110–113.

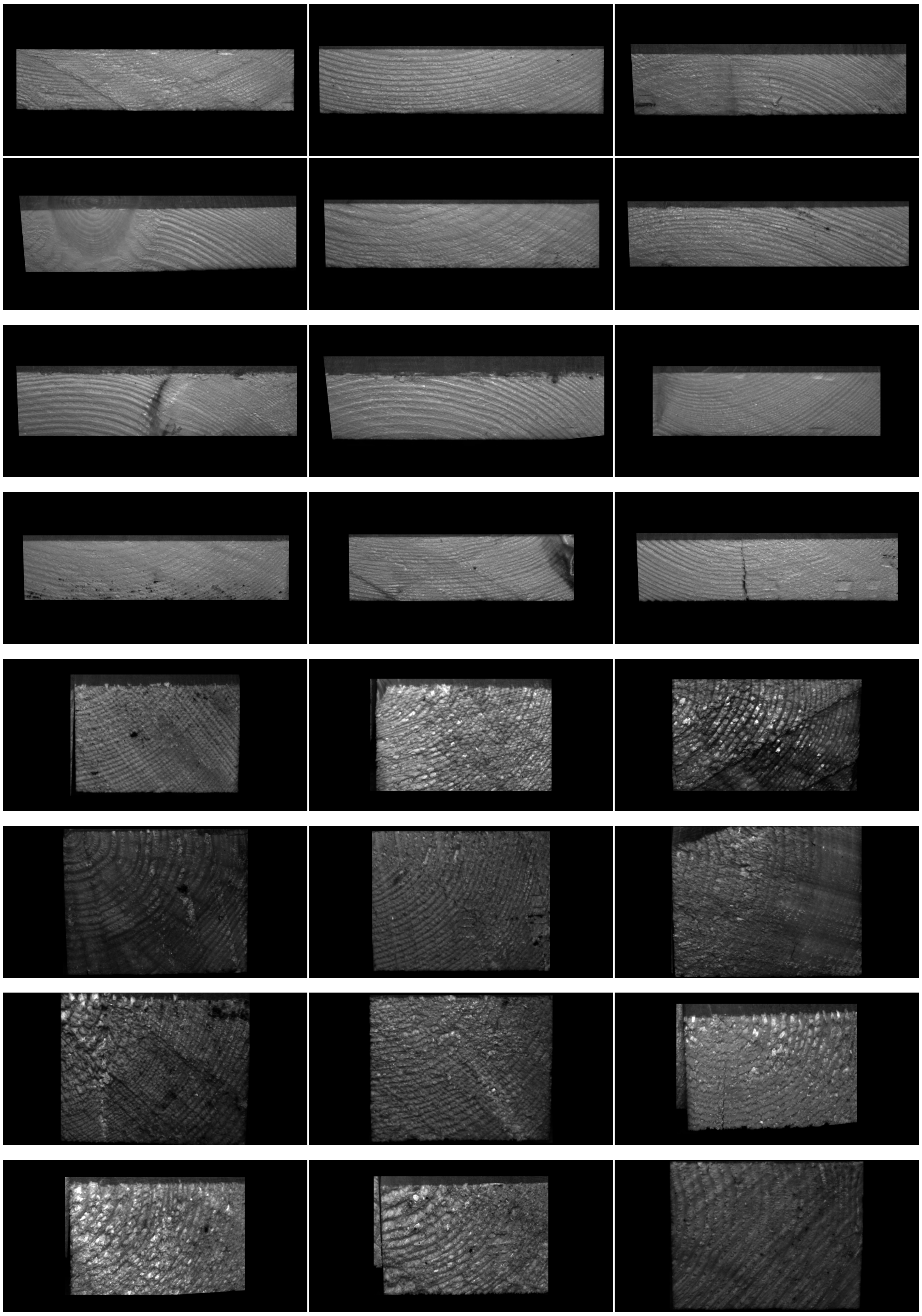
[36] Zhou, Z., Siddiquee, M.M.R., Tajbakhsh, N., and Liang, J. UNet++: A Nested U-Net Architecture for Medical Image Segmentation. *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support (DLMIA)*, Springer, LNCS volume 11045, July 2018, pp. 3–11.

[37] Zuiderveld, K. Constrast limited adaptive histogram equalization. *Graphics Gems IV*, San Diego: Academic Press Professional, 1994, pp. 474–485.

A Appendix A: Data

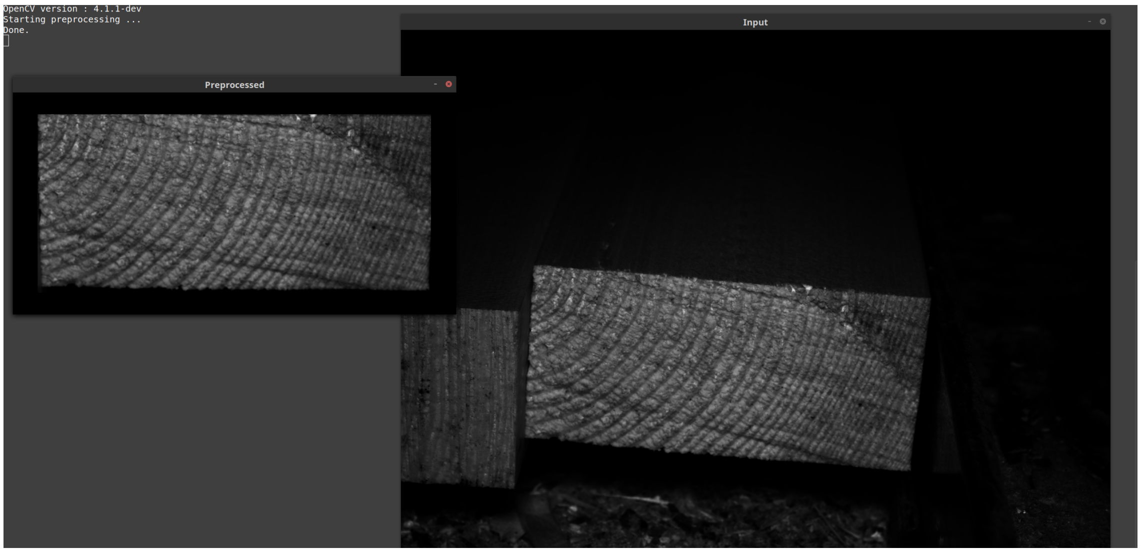
This appendix contains a collage of the data of this thesis; images captured from the factory line with wood end-face extracted. Note the large variability in the wood end-face condition and in the shape of annual rings.





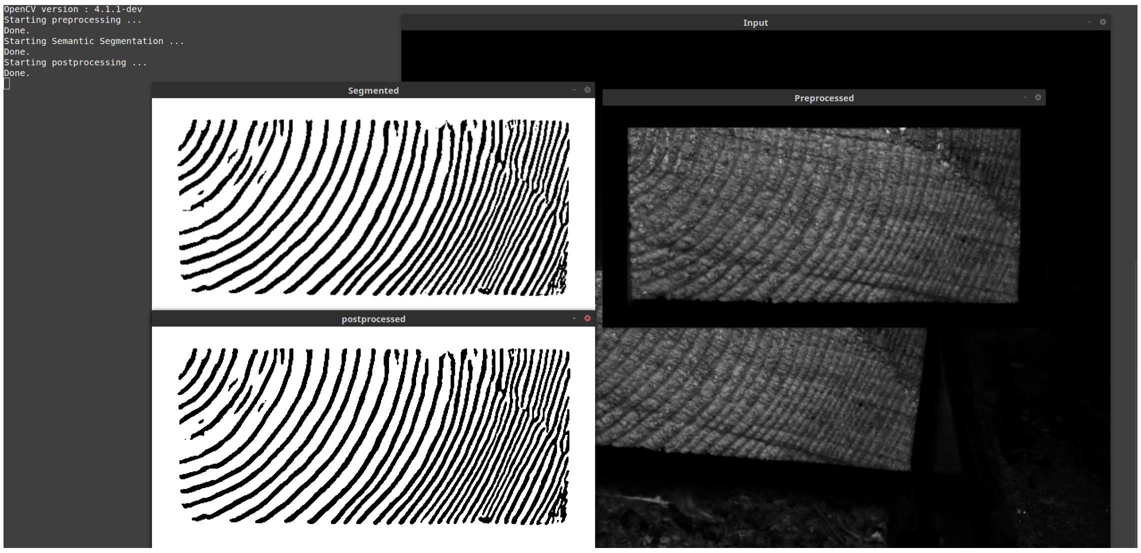
B Appendix B: Example Run of the System

Complete run of the implement annual growth analysis system.



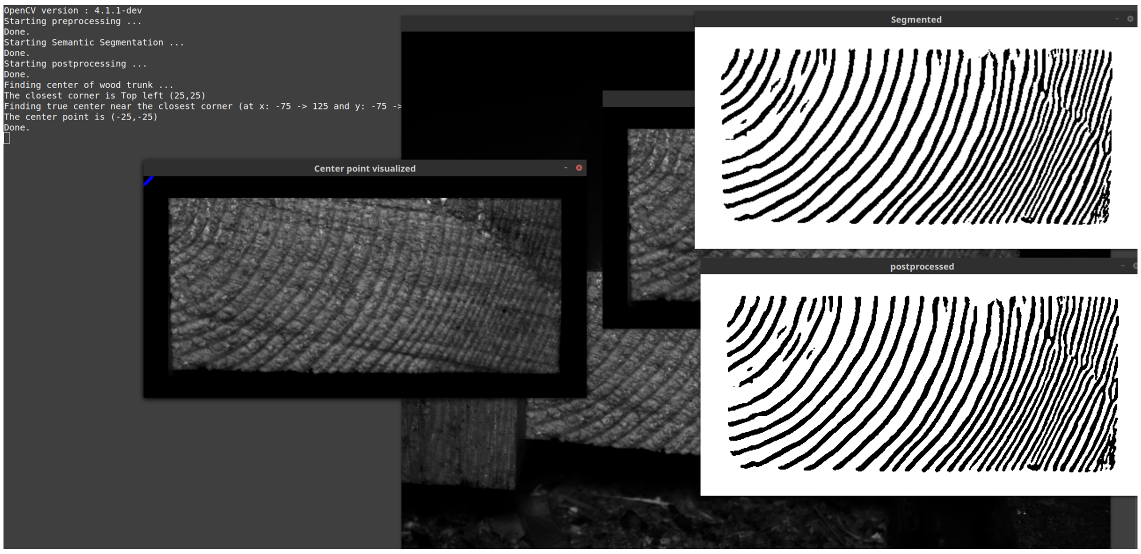
```
OpenCV version : 4.1.1-dev
Starting preprocessing ...
Done.
```

(a) Preprocessing step: wood end-face is extracted and straightened.



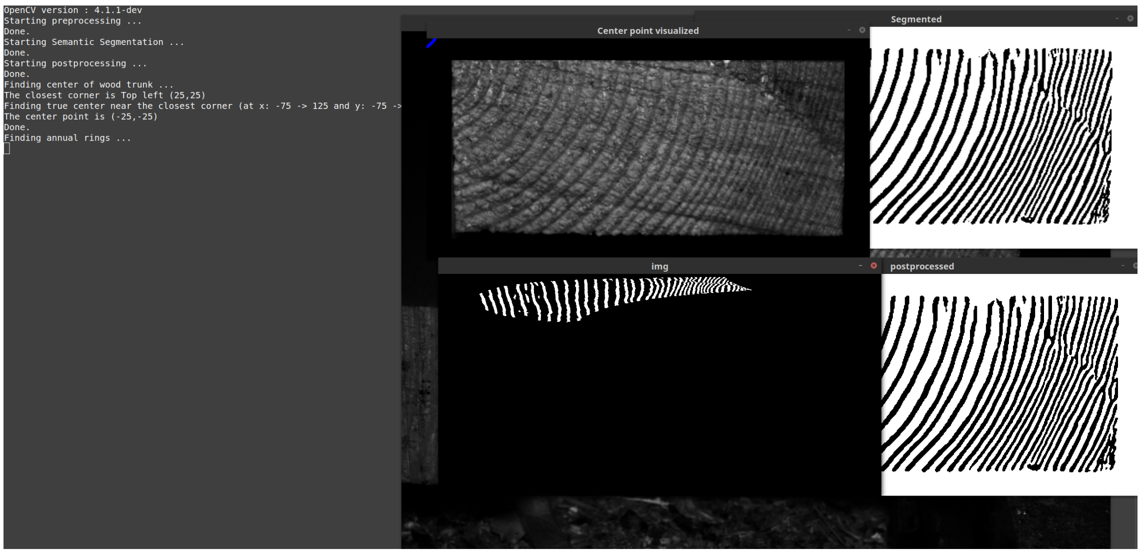
```
OpenCV version : 4.1.1-dev
Starting preprocessing ...
Done.
Starting Semantic Segmentation ...
Done.
Starting postprocessing ...
Done.
```

(b) Annual ring detection step: annual rings are detected using semantic segmentation by a neural network. The output segmentation is further refined by postprocessing.



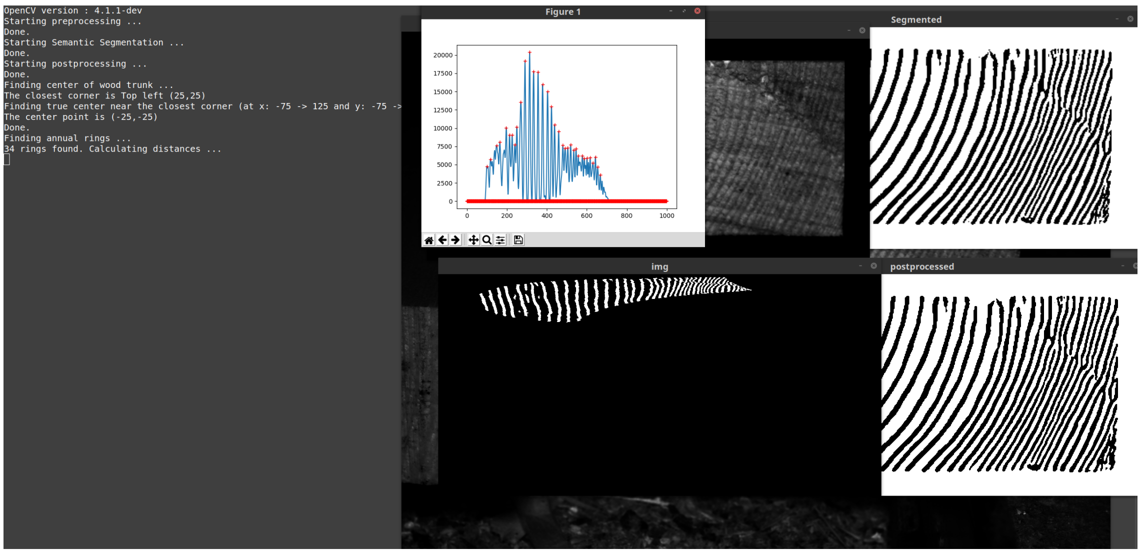
```
OpenCV version : 4.1.1-dev
Starting preprocessing ...
Done.
Starting Semantic Segmentation ...
Done.
Starting postprocessing ...
Done.
Finding center of wood trunk ...
The closest corner is Top left (25,25)
Finding true center near the closest corner (at x: -75 -> 125 and y: -75 ->
The center point is (-25,-25)
Done.
```

(c) Pith detection step: location of the pith is estimated.



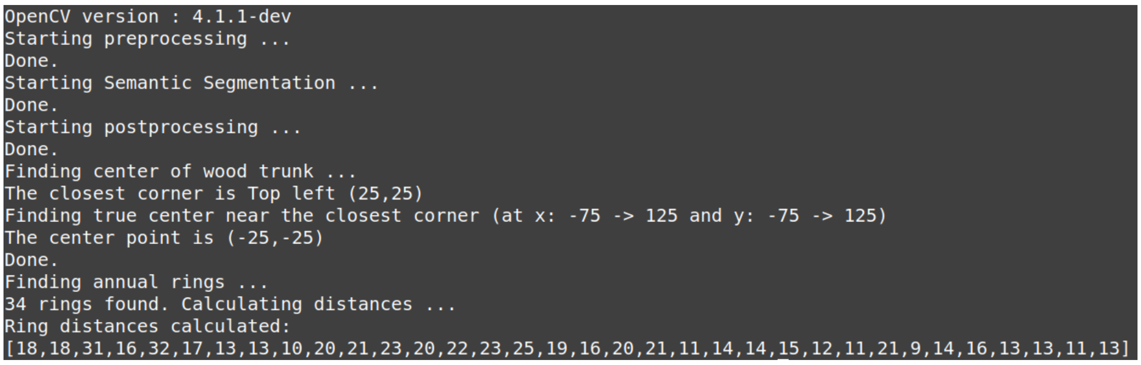
```
OpenCV version : 4.1.1-dev
Starting preprocessing ...
Done.
Starting Semantic Segmentation ...
Done.
Starting postprocessing ...
Done.
Finding center of wood trunk ...
The closest corner is Top left (25,25)
Finding true center near the closest corner (at x: -75 -> 125 and y: -75 ->
The center point is (-25,-25)
Done.
Finding annual rings ...
```

(d) Ring calculation step: image is transformed to polar coordinates at the pith



```
OpenCV version : 4.1.1-dev
Starting preprocessing ...
Done.
Starting Semantic Segmentation ...
Done.
Starting postprocessing ...
Done.
Finding center of wood trunk ...
The closest corner is Top left (25,25)
Finding true center near the closest corner (at x: -75 -> 125 and y: -75 ->
The center point is (-25,-25)
Done.
Finding annual rings ...
34 rings found. Calculating distances ...
```

(e) Ring calculation step: projection is calculated and peaks are filtered.



```
OpenCV version : 4.1.1-dev
Starting preprocessing ...
Done.
Starting Semantic Segmentation ...
Done.
Starting postprocessing ...
Done.
Finding center of wood trunk ...
The closest corner is Top left (25,25)
Finding true center near the closest corner (at x: -75 -> 125 and y: -75 -> 125)
The center point is (-25,-25)
Done.
Finding annual rings ...
34 rings found. Calculating distances ...
Ring distances calculated:
[18,18,31,16,32,17,13,13,10,20,21,23,20,22,23,25,19,16,20,21,11,14,14,15,12,11,21,9,14,16,13,13,11,13]
```

(f) Finally, the annual ring distances are calculated and returned as a vector.

Figure B1: Example run of the proposed wood annual growth analysis system.