TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Environmental Sciences

Automatic Retrieval of Skeletal Structures of Trees from Terrestrial Laser Scanner Data

A thesis submitted in partial fulfilment of the requirements for the degree of Doktoringenieur (Dr.-Ing.)

submitted by
Dipl.-Inf. Anita Schilling
born on 11 March 1985 in Dresden

Reviewers:

Prof. Dr. sc. techn. habil. Hans-Gerd Maas
Technische Universität Dresden

Univ.-Prof. Dr. techn. Norbert Pfeifer
Technische Universität Wien

Sen.-Prof. Dr.-Ing. habil. Uwe Petersohn
Technische Universität Dresden

Dresden, 10 October 2014

Declaration of the doctoral candidate

This is to certify that this copy is fully congruent to the original of the thesis with the topic *Automatic Retrieval of Skeletal Structures of Trees from Terrestrial Laser Scanner Data*.

Anita Schilling
Dresden, 5 November 2014

ABSTRACT

Research on forest ecosystems receives high attention, especially nowadays with regard to sustainable management of renewable resources and the climate change. In particular, accurate information on the 3D structure of a tree is important for forest science and bioclimatology, but also in the scope of commercial applications.

Conventional methods to measure geometric plant features are labor- and time-intensive. For detailed analysis, trees have to be cut down, which is often undesirable. Here, Terrestrial Laser Scanning (TLS) provides a particularly attractive tool because of its contactless measurement technique. The object geometry is reproduced as a 3D point cloud. The objective of this thesis is the automatic retrieval of the spatial structure of trees from TLS data. We focus on forest scenes with comparably high stand density and with many occlusions resulting from it. The varying level of detail of TLS data poses a big challenge.

We present two fully automatic methods to obtain skeletal structures from scanned trees that have complementary properties. First, we explain a method that retrieves the entire tree skeleton from 3D data of co-registered scans. The branching structure is obtained from a voxel space representation by searching paths from branch tips to the trunk. The trunk is determined in advance from the 3D points. The skeleton of a tree is generated as a 3D line graph.

Besides 3D coordinates and range, a scan provides 2D indices from the intensity image for each measurement. This is exploited in the second method that processes individual scans. Furthermore, we introduce a novel concept to manage TLS data that facilitated the research work. Initially, the range image is segmented into connected components. We describe a procedure to retrieve the boundary of a component that is capable of tracing inner depth discontinuities. A 2D skeleton is generated from the boundary information and used to decompose the component into sub components. A Principal Curve is computed from the 3D point set that is associated with a sub component. The skeletal structure of a connected component is summarized as a set of polylines.

Objective evaluation of the results remains an open problem because the task itself is ill-defined: There exists no clear definition of what the true skeleton should be w.r.t. a given point set. Consequently, we are not able to assess the correctness of the methods quantitatively, but have to rely on visual assessment of results and provide a thorough discussion of the particularities of both methods.

We present experiment results of both methods. The first method efficiently retrieves full skeletons of trees, which approximate the branching structure. The level of detail is mainly governed by the voxel space and therefore, smaller branches are reproduced inadequately. The second method retrieves partial skeletons of a tree with high reproduction accuracy. The method is sensitive to noise in the boundary, but the results are very promising. There are plenty of possibilities to enhance the method's robustness. The combination of the strengths of both presented methods needs to be investigated further and may lead to a robust way to obtain complete tree skeletons from TLS data automatically.

KURZFASSUNG

Die Erforschung des Ökosystems Wald spielt gerade heutzutage im Hinblick auf den nachhaltigen Umgang mit nachwachsenden Rohstoffen und den Klimawandel eine große Rolle. Insbesondere die exakte Beschreibung der dreidimensionalen Struktur eines Baumes ist wichtig für die Forstwissenschaften und Bioklimatologie, aber auch im Rahmen kommerzieller Anwendungen.

Die konventionellen Methoden um geometrische Pflanzenmerkmale zu messen sind arbeitsintensiv und zeitaufwändig. Für eine genaue Analyse müssen Bäume gefällt werden, was oft unerwünscht ist. Hierbei bietet sich das Terrestrische Laserscanning (TLS) als besonders attraktives Werkzeug aufgrund seines kontaktlosen Messprinzips an. Die Objektgeometrie wird als 3D-Punktwolke wiedergegeben. Basierend darauf ist das Ziel der Arbeit die automatische Bestimmung der räumlichen Baumstruktur aus TLS-Daten. Der Fokus liegt dabei auf Waldszenen mit vergleichsweise hoher Bestandesdichte und mit zahlreichen daraus resultierenden Verdeckungen. Die Auswertung dieser TLS-Daten, die einen unterschiedlichen Grad an Detailreichtum aufweisen, stellt eine große Herausforderung dar.

Zwei vollautomatische Methoden zur Generierung von Skelettstrukturen von gescannten Bäumen, welche komplementäre Eigenschaften besitzen, werden vorgestellt. Bei der ersten Methode wird das Gesamtskelett eines Baumes aus 3D-Daten von registrierten Scans bestimmt. Die Aststruktur wird von einer Voxelraum-Repräsentation abgeleitet indem Pfade von Astspitzen zum Stamm gesucht werden. Der Stamm wird im Voraus aus den 3D-Punkten rekonstruiert. Das Baumskelett wird als 3D-Liniengraph erzeugt.

Für jeden gemessenen Punkt stellt ein Scan neben 3D-Koordinaten und Distanzwerten auch 2D-Indizes zur Verfügung, die sich aus dem Intensitätsbild ergeben. Bei der zweiten Methode, die auf Einzelscans arbeitet, wird dies ausgenutzt. Außerdem wird ein neuartiges Konzept zum Management von TLS-Daten beschrieben, welches die Forschungsarbeit erleichtert hat. Zunächst wird das Tiefenbild in Komponenten aufgeteilt. Es wird eine Prozedur zur Bestimmung von Komponentenkonturen vorgestellt, die in der Lage ist innere Tiefendiskontinuitäten zu verfolgen. Von der Konturinformation wird ein 2D-Skelett generiert, welches benutzt wird um die Komponente in Teilkomponenten zu zerlegen. Von der 3D-Punktmenge, die mit einer Teilkomponente assoziiert ist, wird eine Principal Curve berechnet. Die Skelettstruktur einer Komponente im Tiefenbild wird als Menge von Polylinien zusammengefasst.

Die objektive Evaluation der Resultate stellt weiterhin ein ungelöstes Problem dar, weil die Aufgabe selbst nicht klar erfassbar ist: Es existiert keine eindeutige Definition davon was das wahre Skelett in Bezug auf eine gegebene Punktmenge sein sollte. Die Korrektheit der Methoden kann daher nicht quantitativ beschrieben werden. Aus diesem Grund, können die Ergebnisse nur visuell beurteilt werden. Weiterhin werden die Charakteristiken beider Methoden eingehend diskutiert.

Es werden Experimentresultate beider Methoden vorgestellt. Die erste Methode bestimmt effizient das Skelett eines Baumes, welches die Aststruktur approximiert. Der Detaillierungsgrad wird hauptsächlich durch den Voxelraum bestimmt, weshalb kleinere Äste nicht angemessen reproduziert werden. Die zweite Methode rekonstruiert Teilskelette eines Baums mit hoher Detailtreue. Die Methode reagiert sensibel auf Rauschen in der Kontur, dennoch sind die Ergeb-

nisse vielversprechend. Es gibt eine Vielzahl von Möglichkeiten die Robustheit der Methode zu verbessern. Die Kombination der Stärken von beiden präsentierten Methoden sollte weiter untersucht werden und kann zu einem robusteren Ansatz führen um vollständige Baumskelette automatisch aus TLS-Daten zu generieren.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Prof. Hans-Gerd Maas for offering me a position in a research project at the Institute of Photogrammetry and Remote Sensing of Technische Universität Dresden. It was the very reason, why I – being a computer scientist – got involved in research on topics related to geo- and forest-scientific topics. In the past four years, I realized that it is an exciting field, and there are still so many problems to explore and questions to solve. In addition, I am grateful for the continuing support during my PhD research.

I also thank Prof. Nobert Pfeifer (Technische Universität Wien) and Prof. Uwe Petersohn (Technische Universität Dresden) for reading and reviewing the thesis.

The core of this thesis was created in the scope of the DFG-funded project PAK 311 "Reconstruction of the 3D forest structure with multisensory methods". I thank our project partners Prof. Sven Wagner and Anja Schmidt (Technische Universität Dresden), and Prof. Oleg Panferov and Dr. Pavel Propastin (Georg-August Universität Göttingen) for the productive collaboration. Furthermore, I would like to acknowledge Prof. Lippold (Technische Universität Dresden), who was always lending his terrestrial laser scanner.

The Eucalyptus data sets were taken in Brazil for the DAAD-funded project PROBAL. I would like to thank our Brazilian project partners at the Department of Forest Sciences of the Federal University of Paraná. Especially, I thank Prof. Christel Lingnau for her persistence and commitment to the project. I am very grateful that she found the time to organize the research stay for Anja Schmidt and me. Furthermore, I appreciate the contributions of the companies Eldorado Brasil and Santiago & Cintra, who assisted our PROBAL project financially, and with equipment and personnel.

For help in practical matters, especially regarding the setup and scanning of study sites, and for the generally enjoyable time at the Institute of Photogrammetry and Remote Sensing, I would like to say thanks to the whole team of the Chair of Photogrammetry. Special thanks go to Anja Schmidt for being a reliable and witty collaborator on both projects, a funny office colleague, and an uncomplicated travel companion to Tharandt and to Brazil.

Ultimately, I thank my parents infinitely much for their constant support. Moreover, I thank my dear sister Tanja for the never-ending stream of constructive and motivating comments.

CONTENTS

1 INTRODUCTION

Forests are the most important terrestrial ecosystem on planet earth. Roughly 31% of total land area is covered by forests of various types [FAO-2010]. Forests are a habitat for countless plants and animals. They play a major role in the atmospheric carbon cycle and thus influence global climate. But above all, forests are closely linked to the cultural history of mankind as living space, and source of food and goods. Up to this day, wood is still a favored material for construction and energy production.

The demand for forest goods, which comprises wood and non-wood products, is high. Moreover, it can be expected to increase further due to the earth's growing population and intensified effects of globalization [FAO-2011]. Hence, large parts of rainforests are quickly turned into plantations for production of bioenergy, pulp for the paper industry, or the food processing industry.

Information about the tree population is fundamental to determine the amount of plants that can be harvested without depleting the resource and thus to manage natural as well as planted forests sustainably [West-2009]. Deforestation is the result of improper management, which has significant negative impact on the socio-economic situation, soil characteristics, groundwater level, and climatic conditions in the area.

Clearly, information about trees is obtained by measuring them. Since the beginning of the 18th century, forest mensuration was treated seriously and became the subject of scientific research [Van Laar-2007]. Today, measurement of dendrometric properties of a single tree is still manual work. As a result, assessment of dendrometric parameters of a larger sample is time-intensive, cumbersome, and therefore costly.

Faster and precise measuring methods are a clear advantage for the forest industry regarding the improvement of harvest forecasting and estimation of yield rates. Moreover, comprehensive data about trees is naturally of high interest to forest science and bioclimatology. But the number of parameters that can be measured in practice at the standing specimen with acceptable effort is limited. Eventually, trees have to be cut down in order to assess properties, which are infeasible to analyze otherwise. Obviously, destructive methods can be applied only once to one and the same plant, rendering long-term monitoring of it plainly impossible.

In particular the crown, which is the tree's interface to the atmosphere, is of major importance to forest research, though only a small amount of measurement data exist [Pretzsch-2011]. Documenting the intricate branching structure of a tree with conventional methods is a highly challenging task, not to mention the foliage that is truly in the focus of research.

The interplay of form and function in a tree crown also inspired Gaudí, who copied the branching structure of trees and based the supporting structure of the cathedral La Sagrada Familia in Barcelona on their model [Tomlow-2002]. It is likewise an impressive example that designing

shapes, which resemble the spatial structure of trees, is by far easier than the accurate reconstruction of an existing real-world tree. Forest mensuration using conventional methods is already reaching its limits. Therefore, the development of alternatives to measure trees automatically receives much attention of forest research and industry.

Instead of performing the work manually, diligent and tedious processing is clearly a task better left to a computer. This requires a representation of the trees that can be processed by a machine. One way to obtain this data is Terrestrial Laser Scanning (TLS). Over a period of about twenty years, TLS has proven to be a very handy tool for freezing a real world scene as a digital 3D point cloud and thus opened up new possibilities of analyzing the world around us.

A terrestrial laser scanner is a device that automatically samples the surfaces of objects that surround it in a sequential fashion. The result is a description of the visible surfaces at a particular moment in time as a high density 3D point cloud. Hence, assessment of objects is translated from on-site measuring to off-site processing in the office. At the same time, the entire process of measuring becomes reproducible.

As shown by Vosselman and Maas [Vosselman-2010] and Sansoni et al. [Sansoni-2009], TLS has already entered numerous application domains and is actively employed for documentation of cultural heritage, as well as in medical analysis, architecture, civil engineering, forensics, and accident assessment. In brief, TLS is used anywhere, where the geometry of real-world objects is of interest.

Quite early, the potential of TLS in forestry has been realized and it has been continuously investigated in numerous studies. The non-destructive character is an especially attractive feature of this remote sensing technique. In addition, once acquired data can be stored and reused, which allows long-time observation of the same plants that was not practicable before to the same extent.

Beyond automating established methods to measure dendrometric properties for forest inventories, researchers consider to leverage TLS also in other areas of forestry. For instance, the assessment of wildlife habitats is vital to protect and preserve biodiversity and initial studies to apply TLS in this context have been conducted with populations of bats and birds [Azmy-2012; Michel-2008]. Furthermore, the analysis of the spatio-temporal development of crowns and canopies is relevant not only to silviculture. Detailed data about the spatial structure of trees are an advantage to radiation transfer models [Propastin-2013], wind field models [Bienert-2010], and fire hazard models [García-2011].

The fundamental task that the majority of these problems share is the retrieval of the spatial tree structure from TLS data. Estimation of conventional dendrometric properties is necessarily the starting point for analyzing trees that are captured by TLS. But a thorough examination requires a comprehensive description of the tree shape that provides a basis on which advanced knowledge about standing trees can be inferred.

1.1 Objective

The general aim of the thesis is the retrieval of the spatial tree structure from TLS data, which has not been addressed as intensively in the literature as the estimation of dendrometric parameters.

TLS data is a set of 3D points. Therefore, the recognition of continuous surfaces and object shapes is a challenging task. Moreover, effects that are inherent to the scanning technique, such as noise in the measurements, and seemingly missing data due to occlusions, need to be taken into account. Movement in the scene due to wind during a scan is also a cause for apparently strange artifacts

in TLS data. In fact, there is never a perfect day for scanning forest scenes without any wind or other disturbing influences. Consequently, those effects have to be considered in the design of processing algorithms.

Up to now, the majority of studies has focused on measuring single, possibly free-standing trees. Most often, the selected specimen is the central object of a multiscan consisting of several co-registered scans. As a result, the considered tree is usually densely covered by sample points and represented with a high level of detail in the point cloud. However, this kind of experimental setup poses serious limitations on the prospective applicability of the developed methods.

For the purpose of assessing densely wooded areas in the long run, we concentrate on approaches that are adequate to analyze entire forest scenes instead of previously selected specimens. That means the considered TLS data sets have been acquired with the intention to assess a representative plot of the study site with high stand density. Therefore, trees in the scans are represented with a varying degree of completeness and detail.

Clearly, only the spatial tree structure that is actually represented in the TLS data can be reconstructed at all. That means that the retrieval may be limited to a partial reconstruction depending on the coverage of the tree by scan points. The generation of a complete tree model by synthesizing apparently missing parts in order to obtain a perceptually pleasing result is not our intent.

The objective of the thesis is the retrieval of the spatial tree structure in a compact form as a tree graph. The resulting reconstruction should be as complete as feasible w.r.t. the given data sets. Considering these requirements, we approach the problem from two different perspectives.

One the one hand, we investigate the reconstruction of a tree on the basis of multiscan data sets. The approach is restricted to trees that have been visible in more than one scan. Processing is performed mainly in a voxel space representation of the data. The result is a graph that approximates the shape of an entire tree to a certain degree of completion.

One the other hand, we examine the internal data organization of a single scan, which leads to a reconstruction approach on the basis of image processing tools. The transition of processing a 2D raster to obtaining 3D structural descriptions is achieved with the application of Principal Curves [Kégl-1999]. The result is a partial reconstruction of the tree, but with high level of detail.

Important to realize is that evaluation of results appears to be a difficult problem in itself. As pointed out previously, the basis of processing is the content of the scan data. Therefore, it is not appropriate to evaluate the results against the real-world tree. However, there is no objective ground truth available for the TLS data either. As a consequence, we rely on a human operator to assess the results of computation with regard to the capability of the considered method and the given input data.

Among the remote sensing techniques that are evaluated in forestry contexts in the literature are also airborne laser scanning (ALS), mobile laser scanning (MLS), as well as the full-waveform TLS prototype Echidna, which is especially built for silviculture [Strahler-2008; Hilker-2010]. van Leeuwen and Nieuwenhuis [vLeeuwen-2010] give a comprehensive overview about the domain. In this work, we deal exclusively with data from standard terrestrial laser scanners as the Z+F Imager 5006i or the Faro Focus 3D. Another restriction is that we focus on the aspect of computational geometry and leave assessment of biophysical properties to the forest scientists.

1.2 Outline

First, we shed light on the term skeleton, which is often used in the literature but only seldom defined, in chapter 2. In addition, we give a brief introduction to TLS, before we review existing approaches to obtain dendrometric information up to methods to retrieve the entire spatial structure of trees from TLS data. Finally, a summary of the properties that are required from the prospective skeleton representation and method w.r.t. the input data are presented.

In chapter 3, we state the contribution of the thesis to the question of retrieving skeletal structures from TLS data. Furthermore, the related publications are listed.

The data sets that have been used to conduct the experiments are described in chapter 4. The study sites and TLS instruments are introduced briefly to provide an overview over the TLS data, which had chiefly driven the development of the proposed methods.

The first method to retrieve the spatial structure from a TLS data set of an individual tree is presented in chapter 5. After determination of the trunk centerline as an initial processing stage, the branching structure is recovered from a voxel space representation of the input data. The resulting skeleton is a tree graph that reflects the main spatial features of the tree, which is a cutout of a larger multiscan. First, we explain the sub procedures of the method. Second, we present experiments and results. Last, we discuss the proposed method in detail and provide suggestions for further development.

In chapter 6, we approach the management and processing of TLS data from the perspective of a raster representation as for instance offered by an intensity image of a single scan. Our investigations lead to a novel data container concept that facilities TLS processing. The core of work that is presented in this chapter is a method for tracing the boundaries of connected components from Connected Component Labeling [Shapiro-2001] that includes depth discontinuities. After describing the procedures, we present experiment results and discuss the findings.

The procedures that are presented in chapter 6 lay the foundations for the second method to retrieve spatial structure from scanned trees, which is introduced in chapter 7. The core concept of the method is the idea that the 2D raster alignment of a scan can be immediately utilized to achieve a segmentation of the corresponding 3D point set. Subsequently, the spatial structure can be represented as a set of polylines, which are the result of the Polygonal Line Algorithm by Kégl [Kégl-1999] that approximates a Principal Curve for a given point cluster. First, we describe the method in detail. Second, we present experiment results. Last, we discuss the proposed method and the intricate interrelationship of its sub procedures. The chapter is closed with some ideas for future work.

In the last chapter, we summarize our investigations and give a critical evaluation of the proposed methods. We conclude with some final remarks.

2 STATE OF THE ART

In this chapter, we briefly review the notion of an object's skeleton that originates in the field of image analysis. Afterwards, we give an introduction to TLS and evaluate the applicability of the skeleton definition to TLS data. Then, we discuss related efforts to retrieve the spatial structure of trees from TLS data. Finally, we summarize the demands on the prospective method and skeleton.

2.1 The Medial Axis

Borrowed from anatomy, the term *skeleton* is frequently used to convey the notion of summarizing the significant geometric features of an object as a line graph. Blum [Blum-1967] was the first to introduce this concept and termed the compact representation as the Medial Axis. He employed an analogy to a fire on dry grassland that starts at the entire boundary of the shape in the same moment and smoothly feeds inward. Spots where both fire fronts collide are considered the Medial Axis.

An alternative way of comprehension is the composition of the object boundary by maximal inscribed disks [Cornea-2007] as depicted in figure 2.1. The boundary of a maximal inscribed disk is tangent to the object boundary in at least two points. In a number of cases, those points will be located on opposite sides. Then, the disk center is located midway between them. As evident in figure 2.1, the Medial Axis consists of the set of disk centers, which are centered within the object shape.

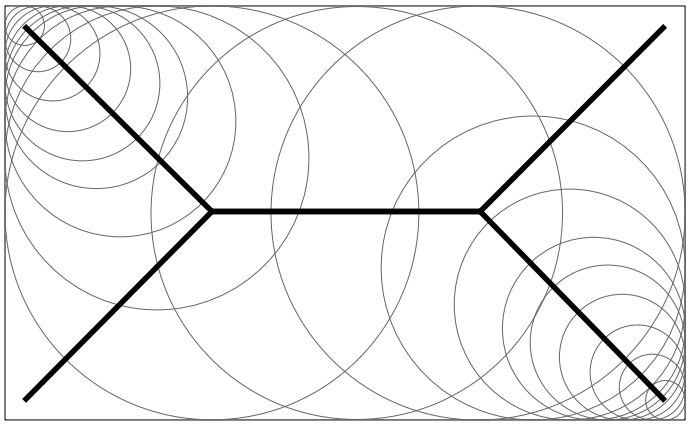


Figure 2.1: Centers of maximal inscribed disk constitute the Medial Axis of the object shape.

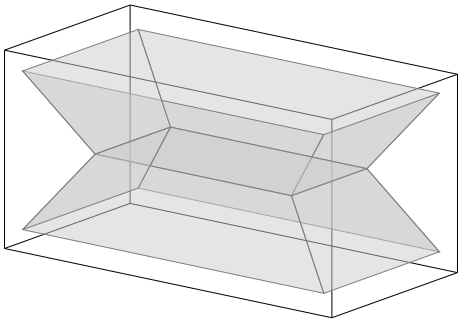


Figure 2.2: In 3D, the Medial Axis becomes the Medial Surface.

The Medial Axis is actively investigated and subject of thorough mathematical analysis because it is a vital concept for questions in numerous fields, such as pattern recognition, and computer vision, but also computer graphics and human computer interfaces [Siddiqi-2008]. The concept that was initially suggested for 2D shapes can be extended to 3D objects. Then, the Medial Axis possibly becomes a Medial Surface [Cornea-2007] as shown in figure 2.2.

In any case, the human reader intuitively grasps the notion of an object's skeleton. Naturally, stick figures share the same fundamental idea. We can easily judge whether a particular stick figure is a good or bad representation of the considered object. Moreover, an average person is usually capable of drawing a skeleton into an arbitrary 2D shape that is perceptually pleasing. Evidently, the skeleton concept is closely connected to human cognition [Siddiqi-2008].

Solutions to compute the true Medial Axis are solely known for simple polygons and polyhedrons [Biasotti-2008]. However, the Medial Axis of complex object shapes is of high interest and motivated plenty of skeletonization approaches that have been proposed since Blum [Blum-1967]. The main tools for this task are Distance Transforms and thinning approaches, the Voronoi Diagram, and Shock graphs. A comprehensive overview about all established methods and aspects of the Medial Axis is given by Siddiqi and Pizer [Siddiqi-2008]. Because the true Medial Axis of complex shapes is exceedingly difficult to obtain in practice, results of the proposed methods are approximations of varying accuracy.

The Medial Axis is very sensitive to geometrical particularities of the shape's boundary as explained in [Siddiqi-2008]. This interrelationship is clearly an inherent feature of the overall concept. But it also means that the Medial Axis reacts to even small modifications of the boundary. Consequently, tiny perturbations in the boundary are reflected by the Medial Axis. For instance, due to a tiny bump in the boundary, an entirely new branch of the Medial Axis emerges as shown in figure 2.3. In addition, the description of an object shape in 2D or 3D is not given in continuous, but in discrete space in the majority of cases. As a result, the boundary of the object may be an approximation of the true shape in itself. Hence, the corresponding Medial Axis contains a number of apparently spurious branches.

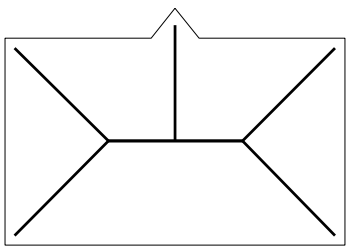


Figure 2.3: Small perturbations in the boundary cause additional edges in the Medial Axis.

When contemplating real-world trees, the expected stick figure intuitively coincides with a tree graph in the graph-theoretic sense. Similarly, the true tree shape can be modeled as a composite of cylinders of decreasing radii. In this case, the cylinder axes roughly correspond to the tree skeleton.

2.2 Terrestrial Laser Scanning

Terrestrial Laser Scanning (TLS) is a ground-based LiDAR (light detection and ranging) technique that rapidly samples visible object surfaces. Since the late 1990s it gained widespread popularity across numerous application domains that benefit from digitizing object geometry such as architecture, civil engineering, cultural heritage, product design, and forensics. For a comprehensive survey on TLS, as well as airborne laser scanning (ALS), we refer to Shan and Toth [Shan-2009] and Vosselman and Maas [Vosselman-2010].

Laser ranging is an active measurement technique: A laser pulse is emitted from the scanning device itself. The first object surface that obstructs the laser's path causes a reflection pulse that travels back to the scanner. A sensor registers the reflection and triggers intensity and range measurement. The amplitude of the reflected pulse relative to the source signal strength is commonly referred to as *intensity*, which we denote by w. Range measurement can be distinguished in two methods:

Time-of-flight (TOF) scanners measure the elapsed time t between pulse emission and reflection registration at the scanning device in order to determine the range d to the object as

$$d = c \cdot \frac{t}{2} \tag{2.1}$$

where c is the speed of light. Clearly, precision of the measurement and the minimal discernible distance are directly dependent on the accuracy of the scanner's internal clock and duration of the query time interval. The maximal measurable range is defined by the signal strength of the laser pulse. Current models by Riegl, for example, can reach up to 6 km [Riegl-2014]. Due to the measurement technique, TOF scanners usually operate slower than phase-shift scanners.

In contrast, phase-shift scanners emit a continuous laser beam. The carrier wave of the laser beam is modulated by amplitude modulation to transport a measurement wave, which usually has sinusoidal shape [Petrie-2009a]. If the laser beam hits an object surface, a beam reflection is registered at the scanning device. By comparing the phase pattern of the source signal to the reflected signal, the phase difference is determined. The phase difference represents a range in the half-open interval $[0, \lambda)$ where λ is the wavelength of the measurement wave. A range beyond this interval cannot be measured with a single measurement wave because the number of full wavelengths between beam emission and object contact cannot be reconstructed. For this reason, several measurements are executed in quick succession and with different wavelengths. Subsequently, the actual range d is determined as the solution of the resulting equation system, as explained by Petrie and Toth [Petrie-2009a]. The maximal resolvable range r_{max} is defined by the measurement wave with the largest wavelength λ_{max}. As a consequence, the range of objects located outside the interval $[0, \lambda_{max})$ is calculated wrongly, since the number of wavelengths λ_{max} before object contact cannot be established. On the one hand, such measurements may be filtered by their particularly low intensity values, since the laser beam strength decreases with distance. On the other hand, such measurements may remain in the final point cloud and appear as noise points. Similarly, the minimal resolvable range is governed by the precision of the phase angle determination. Phase-shift scanners cover comparably short ranges of maximal ca. 100 m, but operate significantly faster than their TOF counterparts.

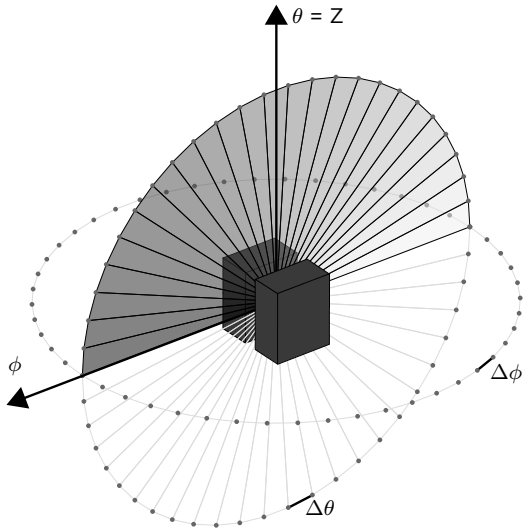


Figure 2.4: The θ axis of the scanner's own spherical coordinate system coincides with the Z axis of the scanner's own Cartesian coordinate system (SOCS). The gray box in the center represents the scanner device.

Regardless of measuring technique, beam divergence is common to all TLS devices. A laser emits highly collimated light, which means that it spreads only minimally with distance. Nevertheless, the area of the cross-section of a laser beam, usually referred to as *footprint*, increases with distance. This property has significant influence on the measuring accuracy, since the laser beam does not sample a geometrically ideal point on the object surface, but in fact a surface area that may cause multiple reflections. Whether one beam return receives priority over the others or all range measurements are combined into one result depends likewise on the strategy of the scanner producer and is commonly not disclosed to the user.

In addition, object material reflectivity, surface slope, and incident angle of laser beams influence the measurement accuracy and may cause noise effects [Boehler-2003; Blaskow-2014]. Furthermore, the authors point out that TLS models are built in small numbers only and the accuracy varies between each instrument.

Besides the applied measuring technique, TLS built types are distinguished by their field of view in full-spherical, panoramic, and camera scanners. Full-spherical and panoramic scanners can sample their entire surroundings with one single scan; whereas camera scanners are limited in view and need multiple scans with different orientation on the same position to cover their surroundings completely. By now, the instruments on the market and their particular characteristics are manifold. Petrie and Toth [Petrie-2009b] give a broad overview about producers and products. In the following, we are concentrating on the Z+F Imager 5006i – a full-spherical, phase-shift TLS device. Another TLS instrument of comparable design is the Faro Focus 3D[1]. Both instruments and the data sets that have been acquired with them are described in chapter 4.

Generally speaking, a typical scan is performed by setting up the scanner onto a tripod. During a scan, a spherical coordinate system with the laser deflection unit as its origin is assumed, as illustrated in figure 2.4. A motor rotates the scanner head in the horizontal XY-plane around its Z axis in discrete steps of $\Delta\phi$. As a result, the horizontal plane is subdivided into $N = \lceil \phi_{max}/\Delta\phi \rceil$ scan lines. At each scan line position, a rotating or oscillating mirror deflects the laser beam in the vertical plane in discrete steps of $\Delta\theta$, resulting in a discrete subdivision of $M = \lceil \theta_{max}/\Delta\theta \rceil$ measurements per vertical scan line. In case of the Z+F Imager 5006i, a scan of the full sphere is

[1] According to [Faro-2013], Trimble recently entered into an OEM agreement with Faro to re-brand and sell the Faro Focus 3D as the Trimble TX 5.

conducted by setting $\phi_{max} = \pi/2$ and $\theta_{max} = \pi$. In this way, the scanner head rotates $\pi/2$ around the Z axis, whereas the mirror performs full turns of π at each increment step. Since the beam deflection unit is a mechanical device, slight deviations of the horizontal and vertical increments occur. That means that the actual observed values of ϕ and θ per measurement are noisy.

In contrast to photography, laser scanning is a sequential measuring technique. A single scan is built up successively by a sequence of measurements sampling the scene in a fixed, predefined spatial pattern. Therefore, a changing scene may cause uninterpretable artifacts in a scan. Especially wind is an important factor in outdoor operation of a TLS instrument. In case of trees, windy weather conditions during a scan causes wavy patterns at branch structures in the point cloud, as demonstrated in figure 2.5. Although, weather conditions should be considered in outdoor scanning projects, we found that in practice there is never a completely windless day and such effects inevitably need to be considered in processing.

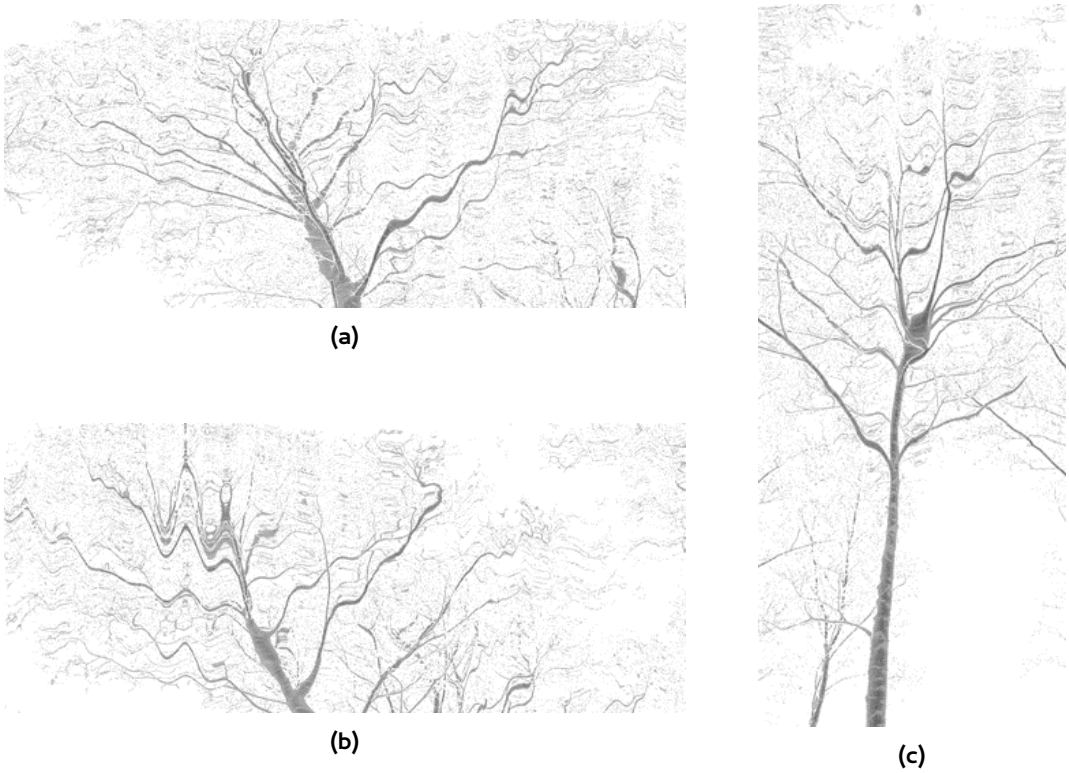


Figure 2.5: Wavy pattern in the intensity image are the result of wind during scanning. The intensity image is inverted, i.e. white indicates zero intensity here.

Internally, the measuring process is performed in a spherical coordinate system. But the result of a scan is a 3D point cloud in Cartesian coordinates with the scanner position at the origin, which is denoted the scanner's own coordinate system (SOCS). The XY-plane is parallel to the ground and the positive Z axis points upward. During a scan, the object surfaces that are in the line of sight to the scanner are sampled. For this reason, the object representation in the point cloud is not a complete sampling of the entire object geometry, as indicated in figure 2.6. Analogous to light, the term scan shadow denotes the unobserved area, which lies behind the point where the laser beam was intercepted by a surface closer to the scanner.

Multiple scans from different viewpoints may be performed and co-registered via artificial markers or object geometry with the intention to sample the target object as completely as possible, as illustrated in figure 2.7. Whether complete coverage of the target object is possible at all strongly

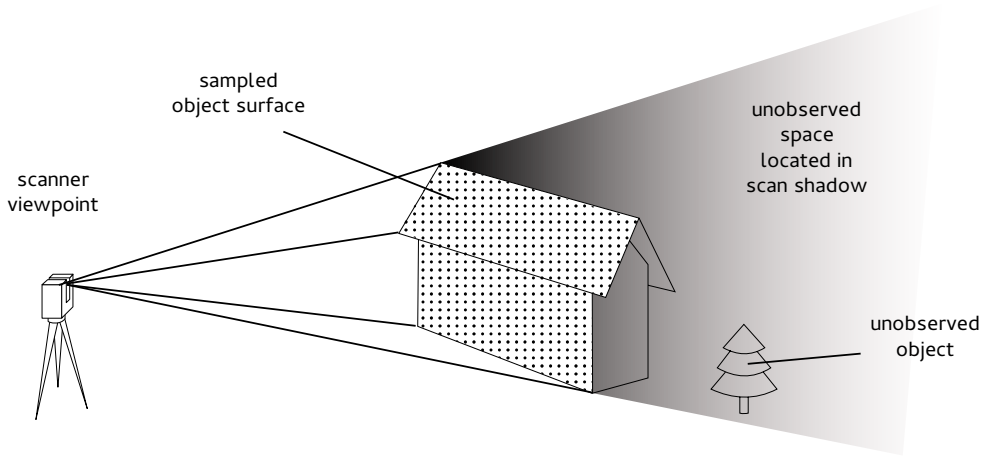


Figure 2.6: Only one side of an object can be sampled from one scanner viewpoint. Object surfaces that are obstructed or that do not face the scanner remain unobserved.

depends on the nature and the size of the object. Clearly, the more complex an object is, the more scans are necessary for complete coverage, or the less complete the result will be respectively.

Co-registration transforms all scans into the project coordinate system (PCS) based on a set of recognizable, common points over all scans. Often, ball-shaped targets are placed in the scene prior to scanning, which are later identified semi-manually in the scans. Afterwards, the TLS software fits a sphere to the 3D points that represent the target surface in order to obtain a defined 3D point with higher accuracy for the co-registration. In most cases, one scan is used as the anchor and all other scans are transformed relative to it. Therefore, the PCS is often identical to one of the SOCS of the scans.

In the standard case, the product of a single scan is a set of points comprising all valid measurements. A measurement is invalid if no pulse reflection returned to the laser scanner. This happens for instance, if the laser beam is emitted towards the sky. For each valid measurement, 3D coordinates (X, Y, Z), intensity (w), range (d), and spherical coordinates (ϕ, θ) are provided by the scanner software. Whereas only spherical coordinates can be obtained for invalid measurements.

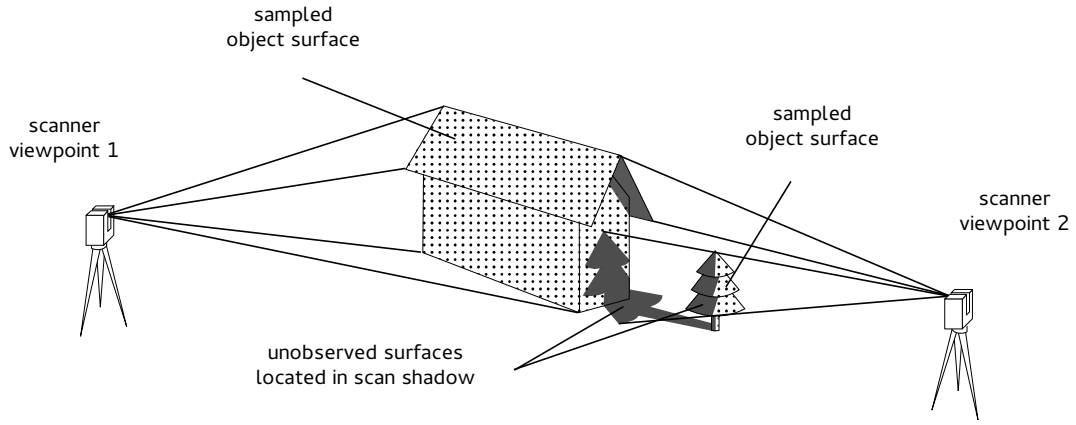


Figure 2.7: The object can be captured from all sides by a multiscan setup. But even if the object is scanned from more than one viewpoint, unobserved surfaces might remain.

During a scan, the TLS data is commonly stored in a proprietary, closed file format. Consequently, operation of a TLS instrument is depending on the same vendor's software to retrieve the measurement data in a file format that is accessible to the user. Though, software export features and therefore actually available measurement data strongly depend on the particular instrument vendor. Unfortunately, there is no definite standard exchange format for terrestrial laser scans between software. In the majority of cases, simple plain text exports of point coordinate lists are used because they are human-readable and easily processable [Huber-2011].

With the power of modern laser scanners even single scans can reach well over 2 GB when exported as plain text file. Considering a project with several, co-registered scans, i.e. a multiscan, this poses a serious challenge for current TLS data handling strategies as well as processing.

2.3 Retrieval of Spatial Tree Structure

In essence, known approaches to estimate the Medial Axis of a scanned object cannot be applied to 3D point clouds obtained by TLS. The prerequisite to compute the Medial Axis is a boundary curve or surface, which is closed. This constraint is not fulfilled by an unordered set of discrete 3D points in continuous Euclidean space.

Naturally, a single scan alone does not contain sufficient geometric information because at least the opposite object half is not visible from the scanner. But also a multiscan may not capture each and every geometrical shape feature. Especially in forest scenes, unobserved surface regions caused by occlusions from other trees and self-occlusions in the crown are inevitable.

A point cloud is a mapping of the true object geometry as a discrete set of points that may be a composite of more than one scan. Therefore, the quality of the mapping of the object shape also depends on the accuracy of the co-registration. A co-registration of poor quality most likely results in a skeletal representation of equally poor quality.

Depending on the tree species, wind in the tree crown causes the upper trunk to bend, which is then visible in the point cloud as displacement. As a consequence, the representation quality of a tree point cloud from a multiscan that was performed under windy conditions is poor in spite of possibly high accuracy of the co-registration.

Clearly, a TLS point cloud usually represents the tree object surface incompletely and with a certain amount of noise. For this reason, approaches that are different from known methods for the Medial Axis have been investigated in studies to retrieve the spatial structure of a tree from TLS point clouds.

The first studies about usage of TLS in a forest were conducted by Hopkinson et al. [Hopkinson-2004] and Watt and Donoghue [Watt-2005], who demonstrated the potential benefit of the instrument in forestry. Since then, the general aim has been to automatize the processing. However, so far a great amount of work is still performed manually or semi-manually. Our focus is on automatic methods that require at most manual input of initial parameters. A clear distinction between methods targeting the determination of selected dendrometric parameters and methods aiming at the retrieval of spatial shape can hardly be made because the transition between both tasks is usually rather smooth. Therefore, we first review approaches that concentrate on dendrometric parameters briefly before we zoom in on methods specifically designated to retrieve the branching structure of a tree.

Retrieval of dendrometric parameters

Dendrometry (from Greek *dendron* = a tree, and *metria* = to measure) covers the measurement of the geometrical properties of trees. The most important parameter is the diameter of the tree trunk at breast height (DBH), which has only a rough international standard. According to [West-2009], the breast height is 1.3 m from the ground at the tree foot point, but can vary up to 1.4 m between countries. In the USA, DBH is defined as 4′ 6″ = 1.3716 m [West-2009].

In order to conduct a measurement relative to the ground surface like measuring of DBH or total tree height, a reference surface is required. An important topic in processing laser scanning data is therefore the determination of a digital terrain model (DTM). The basic assumption is that the ground surface is sampled by a large number of points. Often, the total area in the TLS data is partitioned into square or cubic regions. The DTM then comprises the points with the lowest height coordinate per region after filtering steps are applied [Simonse-2003; Li-2010; Thies-2004; Moskal-2012]. Alternatively, a histogram of point numbers is established along the up direction of each region and the bin with the largest number of points denotes the local ground surface [Maas-2008].

In a large number of studies dendrometric parameters were determined with the DTM as prerequisite, e.g. [Aschoff-2004a; Thies-2004; Maas-2008; Moskal-2012]. Trunk properties can commonly be retrieved with high accuracy if the trunk is covered sufficiently with scan points. Naturally, the trunk is the tree part that is represented by a comparably high number of points in TLS data because it often provides the largest patch of unobstructed, continuous surface area of the considered tree that is visible from the scanner.

The detection of trees present in TLS data is the necessary first step for the assessment of specific geometrical properties. The core of the proposed methods is the determination of candidate point subsets of trunk samples for fitting of circles or cylinders [Maas-2008; Liang-2012; Simonse-2003; Schilling-2011a] or the clustering of points from a particular height slice into disjoint subsets fulfilling certain constraints [Király-2007; Li-2010]. The trunk shape is reconstructed with varying level of detail.

Tree detection and trunk modelling often go hand in hand. Originally, Simonse et al. [Simonse-2003] proposed to utilize the Circle Hough Transform (CHT) by Duda and Hart [Duda-1972] to detect trees. The key idea of the CHT is outlined in figure 2.8. In a horizontal slice through a point cloud, point clusters of tree cross-sections are recognizable by their characteristic circle- or arc-like configuration as shown in figure 2.9. The CHT is a suitable procedure to detect them and was applied by Aschoff et al. [Aschoff-2004b], and Chmielewski et al. [Chmielewski-2010]. As pointed out by Chmielewski et al. [Chmielewski-2010], the assumption that trunk cross-sections are perfect circles usually does not hold. As a result, the performance of the CHT degrades when applied on trees that have more irregular shapes in cross-section or when the TLS data is rather noisy.

A common approach to trunk modeling is the decomposition of the point cloud into slices parallel to the XY-plane, which are evaluated separately. Aschoff and Spiecker [Aschoff-2004a] applied the CHT to detect trunk cross-sections in the point subsets. Detected trunk circles are sorted by height and employed as the starting point of a more sophisticated reconstruction of the trunk surface as a triangulated irregular network. In contrast, Maas et al. [Maas-2008] inspected the point subset of a slice with a mask in order to detect arc-shaped clusters. A circle is fitted to a candidate cluster. Apart from circles, cylinders are frequently fitted to point clusters to retrieve the tree structure as a composite of geometrical primitives. Pfeifer and Winterhalder [Pfeifer-2004b] retrieved tree trunks as a sequence of cylinders that closely approximate the actual trunk shape. Moreover, cross-sections are modeled as B-Splines, which are eventually combined to a spline surface that covers the entire trunk.

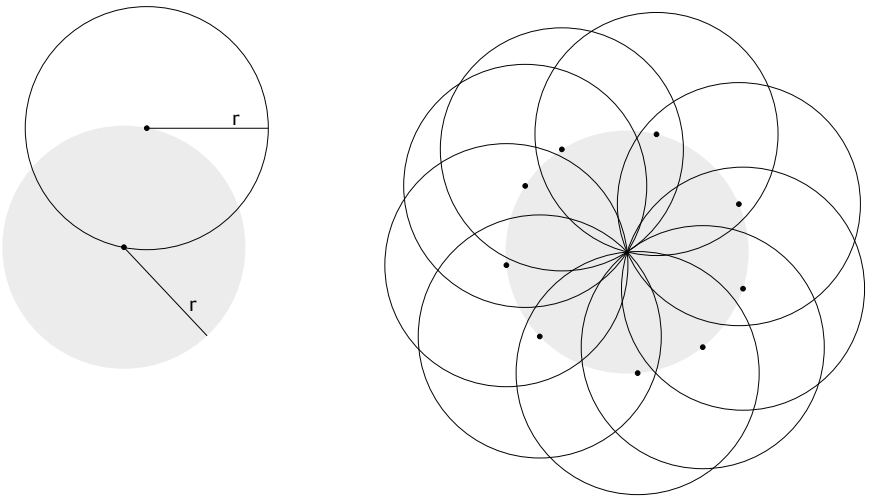


Figure 2.8: Key idea of the Circle Hough Transform. (a) A circle around a point on the perimeter on the sought circle of same radius intersects the sought circle's center point. (b) Consequently, the circle center can be recognized as the spot where all the circles around perimeter points intersect.

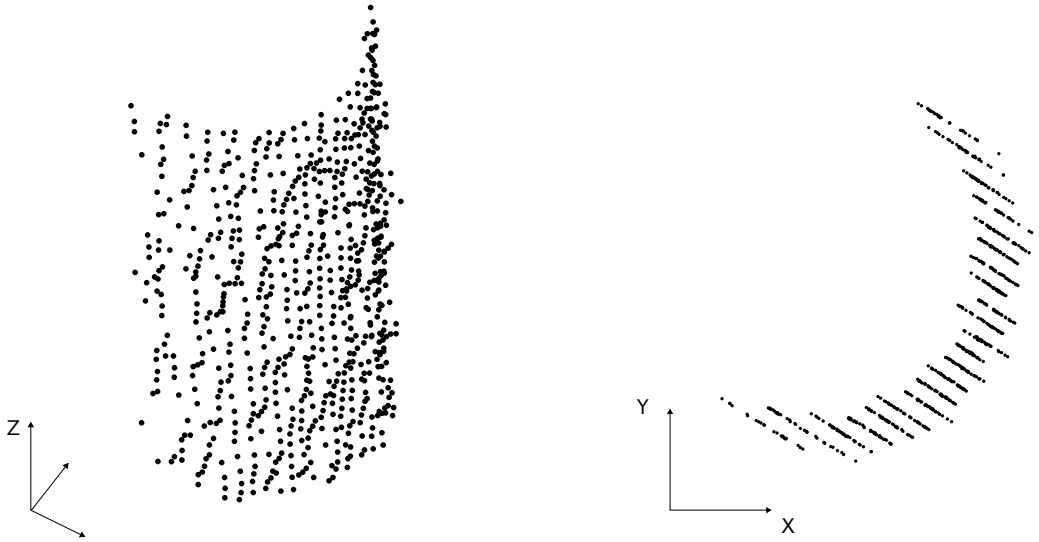
If the trunk shape is retrieved as a set of circles or cylinders, the trunk centerline can be represented as the set of circle centers or cylinder axes. From this information, a 3D polygonal line (polyline for short) that approximates the Medial Axis can be interpolated. In other words, if the trunk is sufficiently covered with scan points, the obtained trunk centerline can be considered to be a part of the skeleton of the tree. However, if the tree representation is not adequate, retrieval of the trunk centerline may not be a trivial task.

Accurate reconstruction of the tree trunk is primarily interesting for the forest industry with the objective to provide precise harvest forecasting, whereas obtaining insights about tree crowns and their dynamics is a central research question in forest science. In comparison to the tree trunk, the crown exhibits highly complex structure.

A first step to assess the crown is the determination of an outer hull from 3D points. Zhu et al. [Zhu-2008] used alpha shapes, which are based on a 3D Delaunay triangulation of the points, in order to carve out the outer hull as a triangle mesh. In contrast, Pfeifer et al. [Pfeifer-2004a] computed a set of polygons that describe the convex hull of crown points in slice-wise partitions. Fleck et al. [Fleck-2007] also determined the convex hull of the crown points after projecting them onto the XY-plane. The proposed strategy is more closely related to established practices to determine the crown spanning area. In general, the description of the crown shape with an outer hull may be useful for biomass estimation in canopies, but does not provide information about the intricate branching structure.

Volume estimates of trees are often performed on a voxel space representation of the data [Moskal-2012; Lefsky-2008; Vonderach-2012; Hosoi-2013; Bienert-2014]. The 3D points are mapped to a constant partition of continuous Euclidean space into cells – the voxel space. The number of voxels that complies with the defined constraints is established and converted to a volume unit with a predefined conversion factor. Clearly, the incomplete representation of the tree is a problematic issue for those procedures.

A related task is the determination of voxel occupancy. For this analysis, laser rays, which are spanned by a 3D point and the corresponding TLS viewpoint, are traced. Traversed voxels are annotated whether they are passed by a beam, not traversed, or occupied, which means that they



(a) front view **(b)** top view

Figure 2.9: Characteristic arc-like point configurations formed by 3D points sampled on the trunk surface.

contain a 3D point. Bienert et al. [Bienert-2010] conducted an analysis to provide data for a wind field analysis. Henning and Radtke [Henning-2006] estimated voxel occupancy parameters to assess parameters pertaining to plant surface. Durrieu et al. [Durrieu-2008] proposed to utilize a constant partition of the spherical coordinate system as voxel space, which is more suited for analysis with regard to the original scan situation.

Retrieval of the spatial tree structure

There are only few approaches in the literature, which determine the spatial tree shape with higher level of detail based on a voxel space representation. Bucksch [Bucksch-2011] mapped the point cloud to an Octree. In this method, the cell-dual of the Octree is constructed: Occupied cells are represented as vertices. If two cells share a common cell wall, the two corresponding vertices are connected by an edge. The resulting graph is restructured based on predefined rules to eliminate redundant edges. Initially, the centroid of the 3D points contained within the considered Octree cell is attached to the corresponding vertex. In the course of processing, vertex coordinates are updated accordingly whenever a substructure of the graph is collapsed. A predecessor of this method was presented by Bucksch and Lindenbergh [Bucksch-2008].

Although, an Octree is chosen as data structure, the inherent hierarchical partitioning capabilities were not exploited by Bucksch [Bucksch-2011]. Instead, the Octree depth is equal for all leaves, which turns the Octree into a common voxel space representation. Bucksch et al. [Bucksch-2010] reported that the procedure is specifically designed for botanical trees. But the test data consists of only three trees of comparably small height; two of which were captured by a multiscan. Nevertheless, the resulting skeleton graph of one of the test trees contains a cycle that is caused by unresolvable noise problems [Bucksch-2010].

Gorte and Pfeifer [Gorte-2004a] proposed an approach to obtain the spatial structure as a graph based on thinning. The point cloud is mapped to voxel space, where trunks appear as hollow tubes if covered sufficiently with scan points on the entire surface. Holes are filled by morphological closing with a predefined structuring element. Subsequently, the component in voxel space is thinned, i.e. layer by layer is removed until no further voxels can be eliminated without changing

the topology of the object. The voxel-dual of the remaining voxel cluster is then constructed as graph, similar to [Bucksch-2010]. Due to ramifications of the scanned tree, ambiguities arise. That means, the voxel cluster might not be exactly one voxel wide in all places. As a result, the corresponding graph contains cycles, which are eliminated by computing a Minimal Spanning Tree (MST). The root voxel of the MST is selected as the lowest voxel of the considered connected component. Prior to computation, edges are weighted differently depending on whether the voxels, which correspond to the vertices adjacent to the edge, share a voxel wall, border or corner. Finally, unique branch labels are established. Starting at the skeleton voxels, labels are propagated towards the exterior in the original unthinned component either by a distance transform [Gorte-2004a] or by a k nearest neighbor strategy [Gorte-2004b].

Commonly, when mapping a point cloud to a voxel space, several disjoint voxel clusters emerge. Connected Component Labeling (CCL) is a handy tool to identify disjoint groups of connected items. Within the scope of this thesis, we utilize a CCL procedure described by [Shapiro-2001] that targets grid data, which is summarized in algorithm 2.1 for 2D pixel grids. In order to identify connected components of non-empty voxels in 3D voxel space, the function that evaluates whether two adjacent items are connected needs to be suitably defined for the task. For instance, the function isConnected can be defined for adjacent voxels v_i, v_j as

```
Function bool isConnected(v_i, v_j)
        return !isEmpty(v_i) ∧ !isEmpty(v_j)
endFunc
```

In addition, the adjacency that is checked in line 8-19 of algorithm 2.1 has to be augmented to test the 13 previously encountered voxels in the 26-adjacency of $D(x, y)$. Other definitions of adjacency are possible as well.

The method presented by Gorte and Pfeifer [Gorte-2004a] was developed further. Gorte [Gorte-2006] proposed to compute a MST directly from the unthinned connected component and to refine the result afterwards. Each voxel is annotated by the length of the longest path of the MST that passes through it. Subsequently, the neighborhood of each leaf voxel, i.e. voxels where no other path passes through, is examined by a mask of predefined size. If any of the considered neighbors is annotated by a longer path length, the center voxel and its associated path are eliminated. A similar strategy was applied by Gatziolis et al. [Gatziolis-2010].

Another modification by Gorte [Gorte-2006] was the integration of a hierarchical approach. Processing is performed starting with a coarse voxel space. Iteratively, the voxel size is decreased and consequently the resolution of the voxel space increases. The skeletonization procedure is constrained to prefer discovered paths of coarser levels of detail. Only in the case of apparent gaps, results from a voxel space with finer resolution are incorporated.

Intuitively, a tree in the graph-theoretical sense is the expected result if the task is the retrieval of a skeleton from a real-world tree. For this reason, the computation of a MST via the shortest path algorithm by Dijkstra [Dijkstra-1959] is a popular tool to select only a subset of all graph edges that form a tree graph. If all edge weights are equal, then all spanning trees that can be identified are equal in total weight. However, the result may not be intuitively the best solution or perceptually pleasing. A MST was retrieved from the cell-dual of a connected component in voxel space in [Gorte-2004a; Gorte-2006; Gatziolis-2010]. Similarly, a MST was computed from the adjacency graph of point clusters in [Xu-2007; Yan-2009; Livny-2010; Delagrange-2011; Wang-2012].

Selecting the voxel size is a major issue for the voxel space approach because it has significant impact on the overall outcome and quality of the processing results. Therefore, a number of methods focus on the retrieval of spatial shape from the raw 3D point cloud.

Algorithm 2.1 Connected Component Labeling with Union-Find Path Compression [Shapiro-2001]

```
1  procedure CCL(D: matrix of N × M range values)
2      L ← 0_{N×M}
3      i ← 2                                              ▷ Label indices start with 2
4      for x ← 0 ... N − 1 do
5          for y ← 0 ... M − 1 do
6              if D(x, y) > 0 then
7                  S ← ∅
8                  if isConnected(D(x, y), D(x − 1, y)) then
9                      S ← S ∪ L(x − 1, y)
10                 end if
11                 if isConnected(D(x, y), D(x − 1, y − 1)) then
12                     S ← S ∪ L(x − 1, y − 1)
13                 end if
14                 if isConnected(D(x, y), D(x, y − 1)) then
15                     S ← S ∪ L(x, y − 1)
16                 end if
17                 if isConnected(D(x, y), D(x + 1, y − 1)) then
18                     S ← S ∪ L(x + 1, y − 1)
19                 end if
20                 S_min ← min S
21                 if S_min ≠ ∅ then
22                     L(x, y) ← S_min
23                     for all s ∈ S/S_min do
24                         union(S_min, s)                ▷ Join labels via union-find path compression
25                     end for
26                 else
27                     L(x, y) ← i
28                     createLabel(L(x, y))               ▷ Create new label in union-find data structure
29                     i ← i + 1
30                 end if
31             end if
32         end for
33     end for
34     for all i ← 0 ... (N − 1) · (M − 1) do
35         L(i) ← find(L(i))                              ▷ Update labels from union-find data structure
36     end for
37     return L: matrix of labels
38 end procedure
```

Côté et al. [Côté-2011] filtered the 3D point cloud manually based on intensity values. Afterwards skeletonization as proposed by Verroust and Lazarus [Verroust-2000] was performed. Experiments were conducted on four selected coniferous trees that had been captured by multiscans. The skeletonization by Verroust and Lazarus [Verroust-2000] is a level set method. Basically, clusters of points per height slice are built. Then, cluster centroids are used as vertices to construct an adjacency graph based on their mutual distance relations. Also, Dijkstra's algorithm is utilized to identify the shortest paths in the graph. A similar idea was proposed by Delagrange and Rochon [Delagrange-2011].

The skeletal structures that are computed by Côté et al. [Côté-2011] provide an initial estimate, but do not completely capture the significant spatial structure of the tree. For this reason, it is utilized only as a starting point to populate it with synthetic branching structure and foliage. The synthesizing of unobserved phyto-elements is achieved using a method described by Runions et al. [Runions-2007] that is based on a space colonization algorithm (SCA).

Xu et al. [Xu-2007] constructs an adjacency graph from the 3D point cloud by connecting a point to each of the neighboring points within a sphere of predefined radius. Again, Dijkstra's algorithm is applied to each vertex to determine the shortest path to a previously selected root vertex. Subsequently, vertices are binned according to path length and adjacency relations. The actual skeleton graph is then built with the centroids of the vertex bins. The basic approach of the procedure presented by Livny et al. [Livny-2010] is similar. However, Livny et al. [Livny-2010] refine the obtained skeleton estimate in an additional optimization step, which minimizes weights that are assigned to vertices. A weight of a vertex is calculated from the extent of the attached sub-tree and the orientation of adjacent edges. After optimization, the skeleton is globally smoothed.

Similarly, Yan et al. [Yan-2009] built an adjacency graph by creating edges between a 3D point and its k nearest neighbors. K-means clustering is used to partition the data set into point groups. For each of the point groups a cylinder is fitted. If that operation does not yield appropriate results, the cluster is subdivided further and cylinder fitting is repeated for each of the newly created point groups. After this iterative subdivision and fitting scheme, the skeleton graph is obtained from the cluster centroids. Furthermore, the skeleton is refined by fitting B-Splines to the retrieved branches. Wang et al. [Wang-2012] also utilizes a combination of k-means clustering and a kD-tree to partition the point cloud into small clusters. Then, normal directions of points are estimated to find the average plane in which all points lie. Afterwards, skeleton vertices are determined by circle fitting.

In contrast, a contraction strategy is applied on the 3D point cloud in [Su-2011] and [Preuksakarn-2010]. Iteratively, 3D point coordinates are manipulated. Su et al. [Su-2011] proposed to move points along their normal directions. Preuksakarn et al. [Preuksakarn-2010] utilized a contraction scheme described in [Giannitrapani-1999] that is based on moving towards the centroid of the points in the vicinity. As a result, spots of high point density are created and the processed point cloud resembles the expected skeleton. Since the number of points remains unchanged, additional processing is required to retrieve a compact, minimal skeleton graph from the condensed 3D point cloud.

Pfeifer et al. [Pfeifer-2004a] approximated the spatial tree shape as a composite of cylinders. Normal directions of all points are computed based on the points in their vicinity. Normals are used to filter stray 3D points and to generate estimates for cylinder axes. A point cluster of tubular shape is reconstructed as a sequence of cylinders. An initial cylinder is fitted to candidate points. In order to determine the next section, the cylinder is slightly shifted along its main axis. The points that are new in its adjacency are used to recompute the cylinder. If the obtained parameters satisfy the predefined quality constraints, the cylinder is accepted and the procedure repeated until a new cylinder is rejected. However, cylinder following has to be initialized manually.

Retrieval of a cylinder composite is also the objective of Raumonen et al. [Raumonen-2013]. They propose to partition the 3D point cloud into small point groups, which are called cover sets. A cover set consist of the points that are located within a sphere of predefined radius. First, the space is randomly covered with sphere centers in order to obtain an initial segmentation. Second, a refined cover set segmentation is generated since a point may be located closer to a different sphere center than the initial one. Then, cover sets, which are oriented parallel to the trunk growing direction that had been computed previously, are identified. From this component, spots where branches are forking off are detected by the distance of associated cover sets to the trunk centerline. An iterative procedure that fits cylinders to point subsets, which are generated by subdivision, is performed in order to reconstruct the branching structure with cylinders. Finally, cylinders that are apparently missing from the composite are added. In other words, gaps between two cylinders, which fulfill certain continuity criteria, are bridged with an additional cylinder.

Recently, Bremer et al. [Bremer-2013] presented an approach for the retrieval of tree skeletons based on region growing. So-called planar surface facets are detected by RANSAC [Fischler-1981] with a model of a 3D plane. Then, found facets are augmented by points of similar normal direction. The step takes the curvature at trunks into account. Subsequently, facet groups having a similar principal direction are merged to segments that are later simplified to 3D polylines.

For the most part, the obtained skeletal structures represent only a fraction of the entire tree skeleton. Though the aspect of creating connections between disjoint skeletal parts is clearly relevant, it is seldom investigated. Wang et al. [Wang-2012] simply searches for the nearest disconnected vertex that complies with the predefined continuity criteria. A bit more intricate is the approach by Xu et al. [Xu-2007], who limit the search to a cone of predefined field of view. Preuksakarn et al. [Preuksakarn-2010] proposed to employ a SCA for this task. Bremer et al. [Bremer-2013] utilizes an iterative strategy related to region growing in order to attach disjoint segments to base segments if their locations and orientations comply with the given continuity criteria.

Beside the voxel space representation and the raw 3D point cloud, a third principal approach to retrieve spatial structure is the range data of a single scan. Most often, the characteristics and origin of the processed range image are not explained. Commonly, it is assumed that the range data is mapped and interpolated to a regular image coordinate system as described by Vosselman and Maas [Vosselman-2010].

Reulke and Haala [Reulke-2005] experimented with a segmentation of the range image based on curvature properties, which are calculated from point groups, to assess forest inventory parameters. The spatial structure of a tree was reconstructed from a range image by Cheng et al. [Cheng-2006]. Normals are estimated with a local shape element to obtain curvature information for points. Then, the data is segmented by considering the similarity of principal directions and range values of points in immediate vicinity. Finally, cylinders are fitted to the point groups and cylinder centroids are connected to form a skeleton graph based on their adjacency relation. The method presented by Dai et al. [Dai-2010] differs only insofar as a quadric surface is fitted to obtain local curvature information and the segmentation is more closely related to a region growing strategy.

An enhanced scheme was proposed by Cheng et al. [Cheng-2007]. First, so-called jump edges are identified. A jump edge results if two neighboring raster elements exhibit a significant difference in range values. Consequently, the corresponding scan points most likely are not located on the same surface patch. The silhouette boundary of the tree object in the range image as well as depth discontinuities between bordering phyto-elements are jump edges, which are used to generate a coarse skeleton. The midpoint between a pair of raster elements, which are located on the same scanline and which are both jump edge elements, is considered a skeleton vertex. The authors report that it is a sufficient approximation of the targeted skeleton. In fact, the approach is obviously closely related to the notion of the Medial Axis as discussed in section 2.1. Second, the set of skeleton vertices is utilized to segment the data into patches. A cylinder is fitted to each patch,

which is used to refine the skeleton vertices. Finally, the skeleton vertices are connected based on their adjacency relation in order to construct a cylinder composite.

Recently, Eysn et al. [Eysn-2013] followed the stick figure idea. Skeletons are manually sketched onto pine trees of an intensity image from a single scan. Then, the associated 3D points in the vicinity of the skeleton are used in a cylinder fitting procedure to generate a cylinder composite of the tree. Clearly, this approach is a straight forward implementation of the skeleton notion, though it is manual work. In this work, we pick up the basic idea of the presented method and specifically address the problem of automatizing the task in chapter 7.

2.4 Issues of the Presented Methods

There are issues that frequently arise in the literature. Above all, the setup design is a crucial factor, which has major impact on the processing strategy as well as the general feasibility of the developed method in research and beyond.

If the focus is on the processing of range and intensity images, then a single scan suffices. However, most methods require the input data as a 3D point cloud. The tree is therefore scanned from several viewpoints to capture it from all sides. The number of viewpoints that are included in a multiscan of a single tree varies between studies. Furthermore, the majority of the conducted experiments concentrate on separate, possibly free-standing trees. In other words, the tree is the sole object in focus of the performed multiscan. Due to the different number of viewpoints, the coverage of the tree can be assumed to vary strongly, as well. As a consequence, experiments that are discussed in studies are not easily comparable or reproducible.

Even though the power of TLS instruments has increased, a scanning project in wooded area is still a time- and labor-intensive endeavor. As a result, the number of data sets that are used to evaluate algorithms is usually rather limited. It is not always stated whether the presented method aims at processing deciduous trees in either leaf-on or leaf-less state or coniferous trees. The retrieval of spatial structure from coniferous trees is evidently an even more challenging problem than the retrieval from deciduous trees in leaf-less state. Similarly, it can be expected that results from deciduous trees with leaves will be less complete than from their counterparts without foliage.

In the long term, taking multiscans of single trees from a representative sample in order to obtain the relevant forest inventory parameters is extremely costly. Despite the development of TLS instruments in the last years and the prospective gain in objectivity and accuracy, such a comprehensive on-site data collection is plainly too time-consuming to compete against conventional methods. For this reason, we focus on processing data of forest scenes. Our data sets are multiscans that do not contain only a single tree, but an entire study area with comparably high stand density. Clearly, a large number of trees are therefore not completely scanned because a high amount of occlusions and self-occlusions occurred.

Naturally, the performance and the result of a particular method depend on the characteristics of the input data set. Often, the input data is expected to represent a tree with a high degree of completeness. As just pointed out, this cannot be guaranteed in a forest scene setup. Consequently, it is not clear how the reviewed methods perform if the data set is a more fragmented representation of a tree.

Within the scope of a research project, new scan data is acquired with the TLS instrument that is at hand. Since there is no strict agreement on the protocol how a scan should be conducted, the outcome of a multiscan is diverse. In addition, the type of the instrument influences the scan result because phase-shift scanners exhibit a different noise characteristic than TOF scanners [Boehler-

2003]. For this reason, it is possible that a particular method performs differently if the test data was acquired with another TLS device. To our knowledge, there is no open, well-documented repository of TLS data of single trees or forest scenes that could be considered for benchmarking the proposed methods.

Moreover, detailed information about the TLS device or scan setup that was used in experiments is not always given. Basically, the discussed methods can be distinguished by the research field they come from: Studies rooted in the environmental sciences usually provide comprehensive information about the utilized instruments, study site, and experimental setup. In contrast, studies originating in computer science commonly propose approaches that are likely to be more computationally complex, but often omit any kind of information about the origin and specifications of the experiment data.

The objective of the presented methods is regularly summarized using the term *skeleton* by the majority of the relevant publications. But as discussed in section 2.1, this term alone does not sufficiently specify what the aim during development of the method actually was. We found that only Bucksch [Bucksch-2011] and Raumonen et al. [Raumonen-2013] actually state requirements for the sought skeleton. Similarly, the initial assumptions, on which the methods are built, are only seldom discussed. As a result, a comparison of methods from the literature is rather difficult due to the different starting points.

Above all, there is no ground truth available that could be compared against the obtained results. Establishing the ground truth of a tree skeleton manually even for a single data set is also a very expensive task. Moreover, ambiguities may arise because data regions might be uninterpretable even for a human operator due to occlusions and noise. In brief, it is exceedingly difficult to objectively quantify the quality of an obtained skeleton. Therefore, the evaluation of the presented method is mainly performed by visual examination of the results. This fact has several implications: The development of novel methods is unintentionally steered by the continuous visual feedback of intermediate results. Consequently, the process is likely biased to produce perceptually pleasing results. Especially if the test data consist of very few trees, the developer runs the risk of tuning the method too much towards the experiment data. However, a "nice" looking tree skeleton is not necessarily the accurate reproduction of what is actually contained in the TLS data set.

The question *What does the TLS data actually represent?* is an extremely difficult one. It is important to realize that the TLS data is a mapping of the real-world tree geometry to a low fidelity representation as a 3D point cloud. If we look at the 3D point cloud, we can usually recognize the object as a tree in spite of data gaps or noise. In our minds, we are able to compose a discrete set of points into a solid object. This happens, because we know how trees ought to look and this meta knowledge influences our perception. The study of perceptual organization began in the early 20th century as Gestalt theory and is an active field of research today [Wagemans-2012]. Because we see the real-world tree, we usually expect the result of the skeletonization operation from the TLS data to concur with this mental image. Evidently, this influences how processing results are judged.

Translating this notion of a mental image into an algorithm is a problem in the field of artificial intelligence and beyond the scope of this thesis. Here, an algorithm is ignorant to the mental image the human operator has of the data and will work through any input data given. Consequently, a method can only be evaluated for the performance on the information that is actually contained in the data set regardless of any meta information a human person might add in his mind. Therefore, a result may not correspond to the human expectation of a perceptually pleasing skeleton, yet be an adequate summary of the spatial structure of the input data.

Exactly because the retrieved tree skeleton might appear as inadequate, a fraction of the studies synthesizes apparently missing phyto-elements in order to generate a plausible tree [Côté-2011; Xu-2007; Livny-2010] w.r.t. visual inspection. For applications in the field of visualization and

virtual reality, this is clearly a reasonable solution. However, we consider it inappropriate if the objective of the research is the truthful assessment of the geometric features of the scanned tree.

2.5 Conclusion

As can be seen, the task of retrieving skeletal structures of trees from TLS data is in fact an ill-defined problem. Given the lack of a clear definition of the expected problem solution, we tried to establish a baseline for our research in form of properties that the prospective method and its results should satisfy. We follow the example of Bucksch [Bucksch-2011] and Raumonen et al. [Raumonen-2013] and give a summary of the determined properties:

Automatic processing Manual intervention during the processing must not be necessary. The method should run fully automatic.

Runtime limits The method should produce results in short time. At best, only seconds are needed to obtain the output. On average, a few minutes to generate the skeleton of an entire tree should be adequate. Yet, processing time should preferably be less than a quarter hour.

Computational efficiency Optimization of the method implementation is a problem pertaining to software engineering and beyond the scope of this thesis. Even though the runtime limits should be kept with a prototypical implementation of the proposed method. This implies that an algorithmically elegant solution is favored.

Control parameters The set of necessary parameters to control the program flow should be as low as possible. Parameters should preferably be easy to comprehend and intuitive to control.

Plausibility of results The method should not explicitly synthesize apparently missing features. Similarly, data gaps should not be filled intentionally. Reconstruction of seemingly missing structures is an issue that can be considered to be future work.

Compact representation Skeletal structures that are produced by the method should be represented in a compact form as a 3D line graph. A real-world tree can be represented by a tree graph, which is the targeted kind of graph. But in the course of processing, ambiguities may arise. Therefore, the constraint that the result has to be a tree graph should not be artifically enforced.

Centering Vertices of a skeletal polyline that describes a branch should concur with the centroids of associated point adjacencies. If the point adjacency has a circle-like cross-section, the centroid of the points is truly on the centerline of the branch. If the branch is not completely represented by points, the centroid of the point adjacency is shifted towards the scanner viewpoint. As a result, the centering of the skeletal structure can only be controlled implicitly by the generated segmentation of points and depends on the input data as well.

Evaluation Evaluation of skeletal structures is done based on visual inspection w.r.t. the input data. In addition, the advantages and weaknesses of the method should be thoroughly analyzed.

The listed properties served as a guideline during the research work. Though, the aspect of finding a working solution to the stated problem has been of paramount importance throughout the research. As a consequence, pragmatic decisions were taken during the development, which was clearly data-driven, to achieve this aim.

In this thesis, skeletal structures subsume the notion of a full tree skeleton, as well as a partial skeleton. The meaning of the term skeleton is depending on the particular method and explained in the particular chapter in correspondence with the described method.

3 CONTRIBUTION OF THE THESIS

The contribution of the thesis is two methods to retrieve the skeletal structures of trees from TLS data. The proposed methods are rather distinct in their respective approaches to the problem. Therefore, the corresponding results also have rather complementary properties.

The first method that is described in chapter 5 targets the retrieval of a full skeleton from a scanned tree. The input data set is a cutout of a larger multiscan and represents an entire tree. The method consists of two main steps.

First, a modification of the Circle Hough Transform [Duda-1972] is proposed in section 5.2.1, which we call Disc Hough Transform (DHT). The DHT is more suited to detect trunk cross-sections from noisy TLS data, which are not of perfect circular shape. After application of the DHT to slices of the TLS data, the trunk centerline can be retrieved as a sequence of center points forming an almost straight line segment via RANSAC [Fischler-1981].

Second, we present an approach to retrieve the branching structure of the tree from a voxel space representation. The trunk centerline is required to mark the corresponding voxels as trunk. A greedy search algorithm is employed to find paths to the trunk from voxels that represent branch tips. The approach is similar to [Gorte-2004a] and [Gorte-2006], but does not require the computation of a MST to eliminate cycles, since no cycles are introduced in the course of processing.

Due to the characteristics of the voxel space, the level of detail of the resulting skeleton is limited. Furthermore, tree positions have to be known in advance. In other words, the input data set is expected to contain a single tree approximately located in the center of the point set. A preceding version of this method has been published in [Schilling-2012b] and [Schilling-2011b].

The second method has been designed with the objective to enhance the level of detail of the tree skeleton in comparison to the results from the first method. For this reason, the method is more complex. The retrieval task is subdivided in two main components:

- The retrieval of boundary information of a connected component in a single scan.
- The retrieval of skeletal structures on the basis of the obtained boundary information.

In chapter 6, Connected Component Labeling [Shapiro-2001] is applied to the range image to partition it into connected components similar to [Bienert-2013]. Boundary tracing as explained by Sonka et al. [Sonka-1998] is adapted to obtain the silhouette outline of the connected components. Here, the key contribution is an extension to boundary tracing that allows the inclusion of depth discontinuities, which is based on a graph-theoretic interpretation of the data that was inspired by Klette and Rosenfeld [Klette-2004] and Gross and Tucker [Gross-1987].

In chapter 7, a novel approach to the task of retrieving skeletal structures from TLS data is presented. The key idea is that a segmentation of the 3D point set can be achieved by analyzing and segmenting the corresponding 2D representation of the data. For this analysis, boundary information is fundamental. After segmentation of scan data in 2D, processing transitions to 3D space. The Polygonal Line Algorithm by Kégl [Kégl-1999] is the essential tool to generate a Principal Curve as a 3D polyline for each of the segmented subset of points, which represents an individual branch. The applicability of the Polygonal Line Algorithm to TLS data has been shown by Schilling et al. [Schilling-2012a].

The basic idea to evaluate the 2D representation of a scan in order to obtain information about the generating 3D object shapes has been recently proposed by Eysn et al. [Eysn-2013]. However, no automatization of the approach has been presented so far. To our best knowledge, the automatic retrieval of skeletal structures as described in chapter 6 and 7 has not been done before in this way.

In summary, both methods run fully automatic. Processing is controlled by a few steering parameters that do not need adjustment between input data sets, which originate from the same multiscan. In other words, the parameters of the procedure do not have to be changed for each individual tree. Therefore, the methods are applicable and suited to process data in batch mode.

Furthermore, the methods are designed to work on data from larger forest scenes. The input data sets are sections from a much larger multiscan, which is described in the next chapter. Therefore, development of the methods has been particularly aimed at dealing with data that exhibits a high amount of occlusions and possibly artifacts due to wind. To our knowledge, this has not been addressed equally intensively yet.

Last, we present a novel approach to the efficient handling and processing of TLS data, which has essential influence on the overall processing scheme that can be applied to scan data in general.

4 DATA SETS, INSTRUMENTS, AND STUDY SITES

In this chapter, we introduce the data sets that are used in the experiments. The TLS data of the study site with birch trees had been acquired prior to the beginning of the actual research work. Consequently, it constitutes the baseline that chiefly steered the development process. Later, another study site with pine trees was scanned with the same TLS device. In addition, a data set representing eucalyptus trees that was scanned with a different TLS instrument is presented. The pine and the eucalyptus data sets are used for testing the approaches presented in chapters 6 and 7. The birch data set is the base data and is used for all the experiments in chapters 5, 6, and 7.

Figure 4.1: An intensity image from data set B. White corresponds to zero intensity.

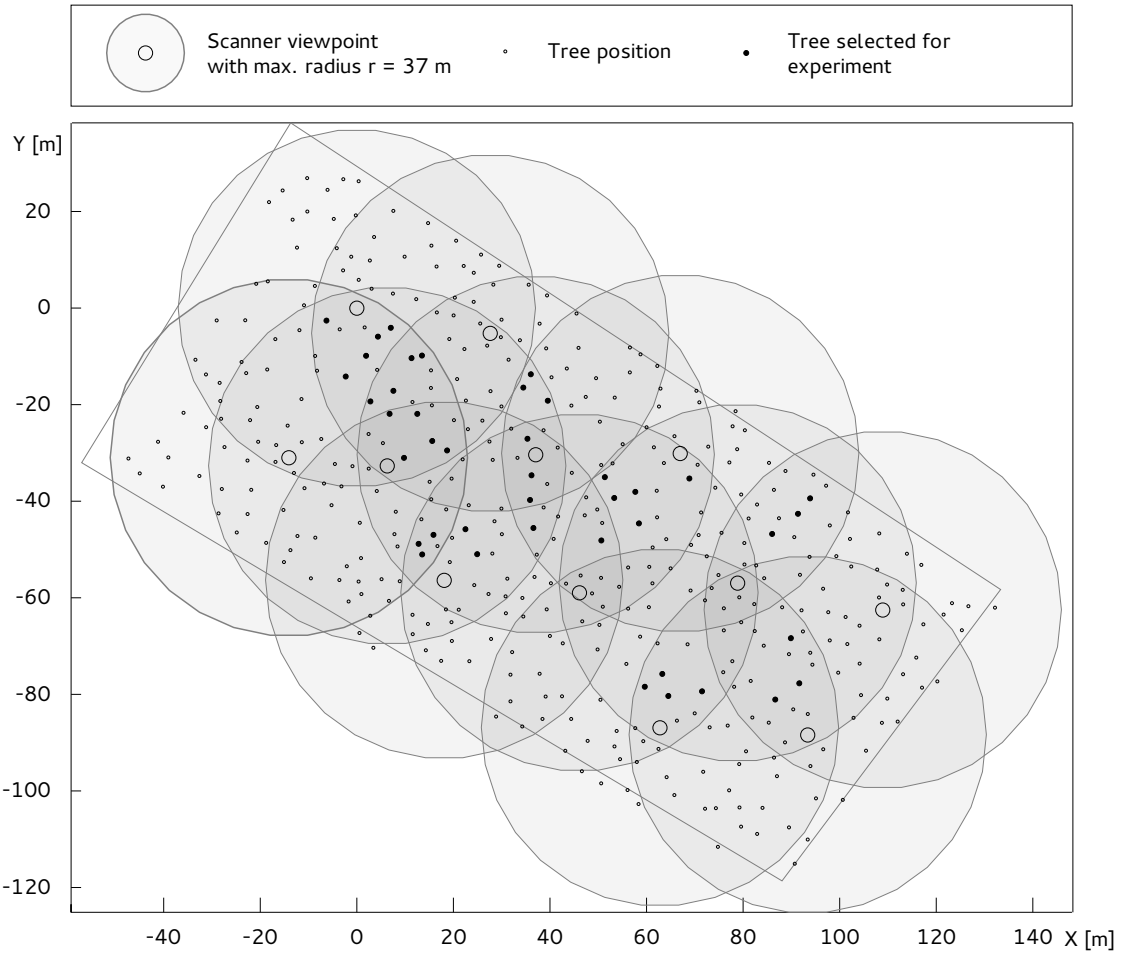


Figure 4.2: Overview of the tree positions and scan setup on the study site where data set B was taken.

4.1 Data Set B – Birch trees

The study site (50° 58′ 37.8″ N, 13° 41′ 53.0″ E) is located in Wilmsdorf near Dresden, Germany, and comprises a plain stock of Silver Birch (*Betula Pendula*). The area is about 1.3 ha (160 m × 80 m) in size. A total of 422 trees including a few conifers were counted within the established rectangular region of interest. The birch trees, which are more than 55 years old, have been under observation by the chair of Silviculture of TU Dresden for more than 40 years. The Silver Birches exhibit a rather straight growth and reach up to about 28.3 m in height. DBH varies between 16.6 cm to 34.4 cm according to manual measurements that were conducted on selected trees on-site with a measuring tape. Mostly, the crown begins well above 10 m from the ground and commonly no branches are forking off in between. Figure 4.1 shows an intensity image of a scan on the study site, which was scanned in the scope of the DFG project PAK 311 "Reconstruction of the 3D forest structure with multisensory methods".

The Z+F Imager 5006i [ZF-2005] was used to scan the study site. 12 fixed viewpoints were marked on the site in order to achieve a sufficient overlapping between the scan regions as depicted in figure 4.2. Each scan was restricted to the measurements that are sampled within a radius of $r = 37$ m around the scanner position. 38 spherical targets were permanently mounted to selected trees for co-registration, which was done semi-manually with the Z+F LaserControl software after scanning. Co-registration accuracy is given in table 4.1. The scans were then transformed to the SOCS of the first scan.

Scanning was accomplished in March 2010, when the birch trees were still in leaf-less state. The TLS instrument was set up on a tripod in about 1.6 m height. For all 12 scans, angular resolution was set to $\Delta\phi = \Delta\theta = 0.018°$. In order to achieve a full-spherical scan, the field of view in the vertical plane was set to $\theta \in [0 \dots 360°]$. Accordingly, the field of view in the horizontal plane was set to $\phi \in [0 \dots 182°]$ to achieve about $2°$ overlapping with the objective to ensure a gap-less scan. On average, each of the 12 scans has about 36.4 Mio. scan points that are valid measurements under the given range constraint.

For the experiments, a set of 42 trees from the study site was selected. Figure 4.2 gives an overview about their positions and resulting scan coverage. The trees were separated automatically such that each one formed an individual data set. A cylinder with radius $r = 6$ m was cut out around the known tree position. The smallest tree data set contains 801,787 points; the largest tree data set has 14,231,724 points. Tree data sets comprise about 4,162,461 points on average. Clearly, the closer a tree is located to a scan viewpoint, the higher the number of points at the tree will be.

Average deviation	Standard deviation	Maximal deviation
12.3 mm	7.8 mm	45.5 mm

Table 4.1: Results of the co-registration of the 12 scans from data set B.

4.2 Data Set P – Pine trees

The study site (50° 58′ 6.1″ N, 13° 33′ 12.7″ E) is located close to Tharandt near Dresden, Germany, and comprises a plain stock of Scots Pine trees (*Pinus sylvestris* L.). Within the area that is about 1.2 ha in size, a total of 552 pine trees were identified. In contrast to leafless birch trees, pine trees have a dense crown with needles that are grouped into small clusters. The crown starts relatively high at the trunk. Before the crown starts, smaller branches may fork off. An overview of a scan is given in figure 4.3. Like the birch study site, the pine trees were scanned in the scope of the DFG project PAK 311.

Figure 4.3: An intensity image from data set P. White corresponds to zero intensity.

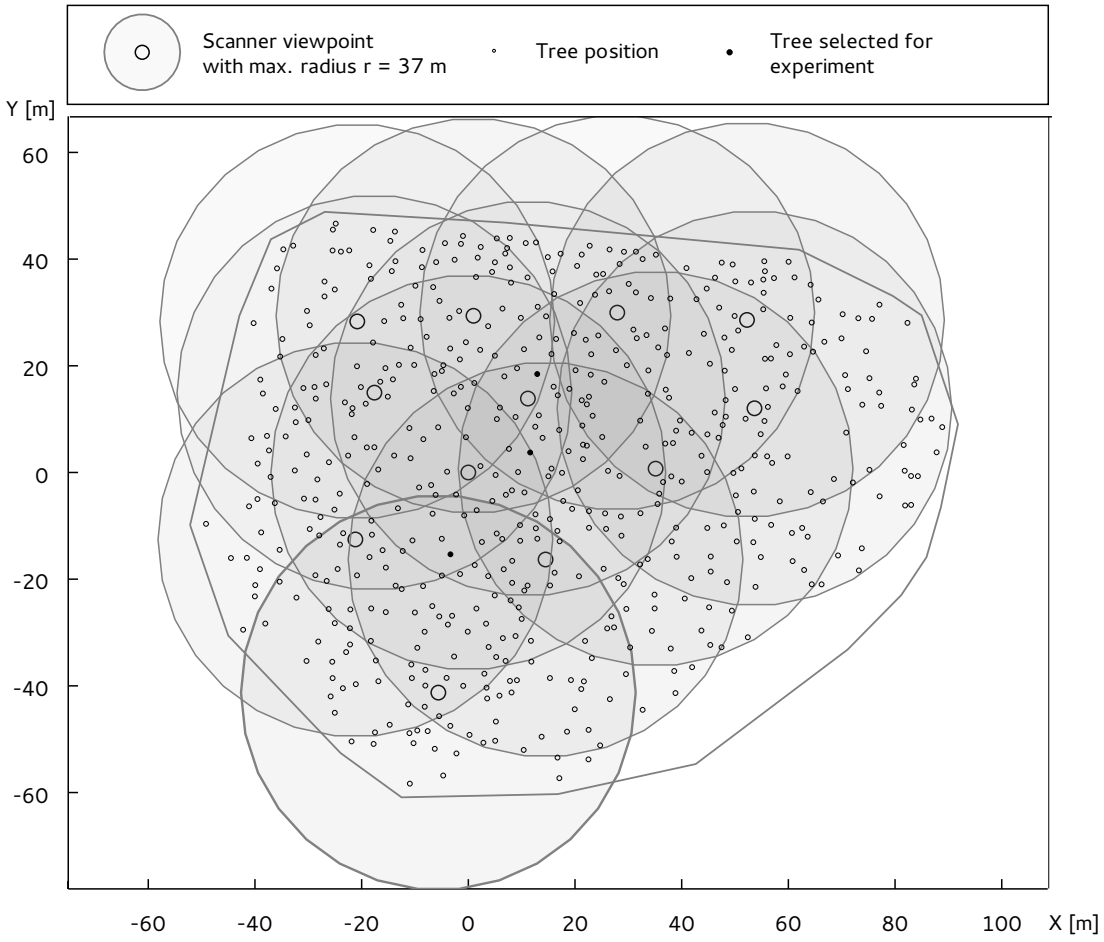


Figure 4.4: Overview of the tree positions and scan setup on the study site where data set P was taken.

The setup and scanning did not differ from the descriptions for data set B: The Z+F Imager 5006i with the same settings for angular resolution was employed. 12 fixed viewpoints were marked permanently and co-registered manually subsequent to the scanning on the basis of 38 spherical targets that had been mounted to selected trees. Co-registration accuracy is given in table 4.2.

For the experiments, we have chosen three pine trees, which are marked in figure 4.4. The pine trees serve as a test to assess whether the approaches in chapter 6 and 7 can be applied to TLS data of other species. In general, the approach aims at processing data of deciduous trees.

Average deviation	Standard deviation	Maximal deviation
8.0 mm	3.8 mm	18.8 mm

Table 4.2: Results of the co-registration of the 12 scans from data set P.

4.3 Data Set E – Eucalyptus trees

The study site is located on the eucalyptus plantation Fazenda Campo Limpo 1 (20° 29′ 4.30″ S, 51° 44′ 42.94″ W) of Eldorado Brasil[1] near Três Lagoas in Mato Grosso do Sul, Brazil. The eucalyptus

[1] www.eldoradobrasil.com.br

trees (*Eucalyptus urograndis*) are approximately fully grown and about 26.5 m tall. The bark of the trees is mostly rather smooth. However, patches of bark can peel off and hang down in shreds that obstruct the actual trunk surface of the tree. The eucalyptus trees have a comparably small crown and little branching structure. Along the entire trunk length, very thin but long branches can fork off. Figure 4.5 shows the intensity image of the scan. The data was captured in the scope of the DAAD project PROBAL in cooperation with the Universidade Federal do Paraná in Brazil.

In the experiments, we include one recording of a eucalyptus plantation that was scanned with the Faro Focus 3D 120 [Faro-2013]. The angular resolution was set to $\Delta\phi = \Delta\theta = 0.035°$. The scan covers a field of view of $\theta \in [0 \dots 152.5°]$ in the vertical plane and $\phi \in [0 \dots 110°]$ in the horizontal plane. In sum, 6,347,505 valid scan points were sampled. The number or position of the individual trees that are located on the study site was not assessed.

The data set of eucalyptus trees is used in the experiments in chapter 6 and 7 with the objective to apply the presented approach to data sets that were acquired with different settings and another TLS instrument. As pointed out before, the scanned data may exhibit differing characteristics depending on the employed TLS device. Clearly, testing with data sets of various TLS instruments and settings is vital in order to ensure that the approach is not tuned too much toward a specific appliance or scan setup, which may severely limit its general applicability.

Eucalyptus trees are not of particular importance to forest research. The plantations are operated with the aim to produce pulp for the paper industry. Therefore, the trunk that provides the largest biomass of the plant is exclusively of interest. In the context of commercial forest management, rapid and accurate assessment of the tree population is of utmost importance in order to assess and confidently predict prospective yield rates and resulting profits.

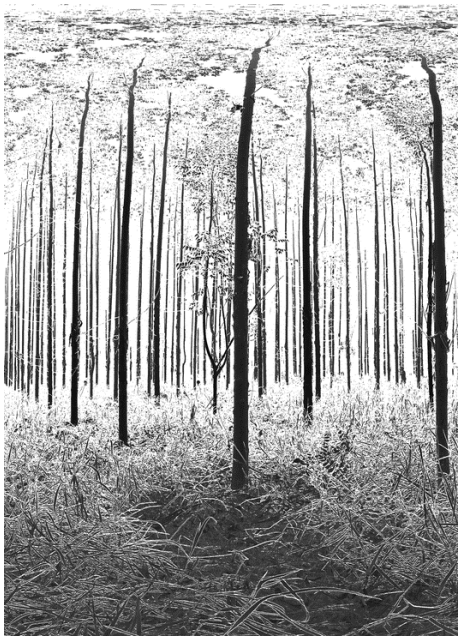


Figure 4.5: An intensity image from data set E. White corresponds to zero intensity.

5 SKELETON RETRIEVAL FROM 3D POINT CLOUDS

In this chapter we introduce a method to retrieve the spatial structure of a tree from its voxel space representation. First, the trunk centerline is determined via a modified Circular Hough Transform [Duda-1972]. Subsequently, the tree graph is obtained by an iterative application of a search algorithm. Finally, we present experiment results and discuss the proposed approach in detail.

5.1 Motivation

Data set B is a co-registered multiscan containing numerous trees. Therefore, we decided to break the larger data set down and focus on cutouts. We consider only a bounding volume of interest with the tree approximately located in the middle. Tree detection can be performed in advance as discussed in section 2.3.

Restricting the analysis to a bounding volume, which contains only a single tree, allows the creation of a voxel space with comparably small voxel size of 10 cm or less. Voxel space analysis is an established tool to process 3D point clouds because it provides a discrete partition of continuous space and thus creates neighborhood relations, which the point cloud usually lacks. For this reason, we have opted for using a voxel space representation in order to facilitate processing.

In data set B as primary example of a larger forest scene, the stand density is comparably high. A tree is usually not scanned from all sides. As a result, the trunk is not represented as a complete, hollow tube of voxels. Yet, trunks are in general well-represented in scans in comparison to the branching structure. The lower parts of trunks are located closer to the scanner than crowns and therefore tree trunks are more densely covered by sample points. As a consequence, they are easier to recognize in 3D point clouds than other phyto-elements. In addition, a number of reasonable assumptions can be made like the average direction of growth, expected diameter, and minimal length that help to recognize tree trunks in the data set.

Hence, we propose to retrieve the trunk centerline from the 3D point cloud by utilizing a variant of the Circle Hough Transform. Afterwards, we begin processing in voxel space in order to retrieve the branching structure of the crown. Naturally, we can expect the crown to be positioned above the retrieved trunk structure, but connected to it. In fact, if a branch of a real-world tree is followed starting from its tip, it is eventually connected to the tree trunk. Similarly, if a data set of a single tree is transformed to voxel space, a single branch will most likely be represented by a chain of adjacent occupied voxels.

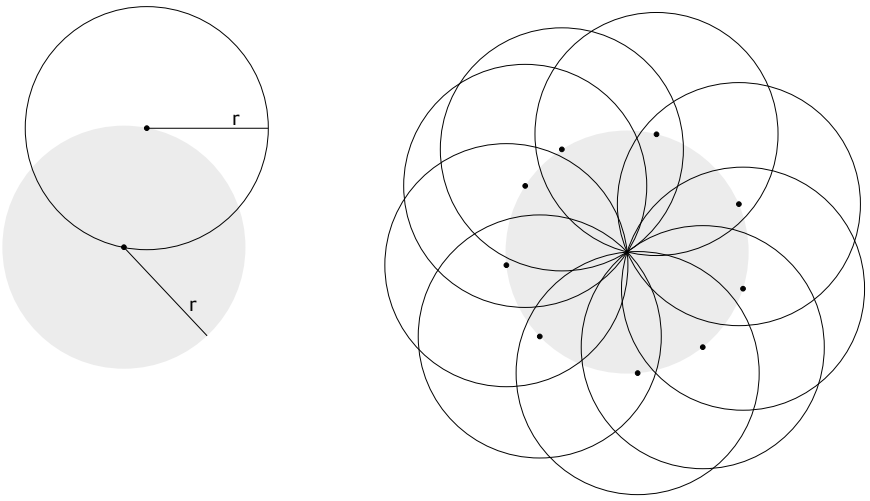


Figure 5.1: Key idea of the Circle Hough Transform. (a) A circle around a point on the perimeter on the sought circle of same radius intersects the sought circle's center point. (b) Consequently, the circle center can be recognized as the spot where all the circles around perimeter points intersect.

Basically, we propose to retrieve the branching structure in voxel space by searching paths to trunk voxels from voxels that are supposedly representing branch tips. For prior identification of branch tips, we weight each voxel with its distance to the closest trunk voxel. On the basis of these weights present in a voxel's 26-adjacency, a suitable set of candidate voxels can be found.

5.2 Method

For the retrieval of tree trunks, we propose a variation of the Circle Hough Transform. Afterwards, the branching structure is obtained from a voxel space representation. The result of the proposed method is an approximation of the spatial tree structure as a graph. Clearly, the skeleton only resembles the actual tree geometry to a certain degree because the level of detail of the provided voxel space is limited.

5.2.1 Disc Hough Transform

The original Hough Transform was developed by Hough [Hough-1962]. Duda and Hart [Duda-1972] revised the original formulation and turned it into a tool to recognize parametric curves in images. The Circle Hough Transform (CHT) denotes the restriction to a circle as a special case of a parametric curve. Figure 5.1 depicts the key concept of the CHT. All circles $c_i \in C$ with radius r intersect in one point p_m if the center points of the set C are located on the perimeter of the same circle m, which has radius r as well. Consequently, the point p_m is the center point of circle m. If the radius r is not known in advance, it can be obtained by repeated application of the procedure with different radii until intersection of all circles occurs.

Although the intersection point of all circles can be determined analytically, it is exceedingly difficult to achieve in practice. In fact, the strength of this tool lies in its efficiency when applied to discrete raster images. Instead of the continuous parameter plane in figure 5.1, a 2D grid A is utilized as the parameter space. Each grid cell functions as accumulator that is initialized as empty.

Let Q denote the set of selected pixels that are not background. Then, a query pixel $q = (q_x, q_y) \in Q$ is transformed to its parametric representation with

$$(q_x - a)^2 + (q_y - b)^2 = r \tag{5.1}$$

The equation holds for a set H of parameter tuples $(a, b) \in \mathbf{Z}^{2+}$. Each parameter tuple (a, b) is a 2D index to an accumulator unit $A(a, b)$ in the parameter space grid. On access, the accumulator value is increased by one. Obviously, if the radius is not known in advance, a 3D accumulator grid can be used. The additional dimension represents increasing radius values. As a consequence, the accumulator with the maximum value is the grid cell, where the maximum amount of circles intersects. The obtained index (a, b) of the accumulator grid concurs with the 2D coordinates of the sought-after circle center in the raster image. Similarly, the radius r may be obtained as third index from the 3D parameter space grid.

Application of the 2D CHT to 3D TLS data requires a pre-processing step to transform 3D points into an appropriate 2D representation. Considering that the growth direction of tree trunks in the data sets is approximately parallel to the Z axis, we partition the bounding volume of the data set along the Z axis into bins of size t_Z. Each bin represents a slice through the point cloud parallel to the XY-plane in SOCS. For each slice, a 2D histogram of point numbers is created. We initialize a 2D grid B, which partitions the XY-plane into cells of side length t_c. The set of 3D points, which are associated with a particular slice, is projected onto the XY-plane and mapped to the corresponding 2D grid. Consequently, each grid cell holds the number of points that project onto it as shown in figure 5.2.

After preprocessing the TLS data, the CHT can be applied. However, results of the CHT are suboptimal on this kind of input data because scan data is rather noisy. The bark surface texture causes additional point displacement from a perfect circle. Moreover, Chmielewski et al. [Chmielewski-2010] pointed out that a trunk cross-section seldom resembles a perfect circle. For this reason, the authors proposed a fuzzy CHT variant.

In order to relax the constraint that the model has to be a perfect circle, we propose to use a disc instead of a circle in the accumulation routine. In this way, the disc radius functions as an upper limit on the unknown radius r and a 3D parameter space grid is not required. In contrast to standard CHT that considers only each filled pixel of the 2D grid B, we consider each 3D point separately in the accumulation process. In other words, we do not summarize points that project

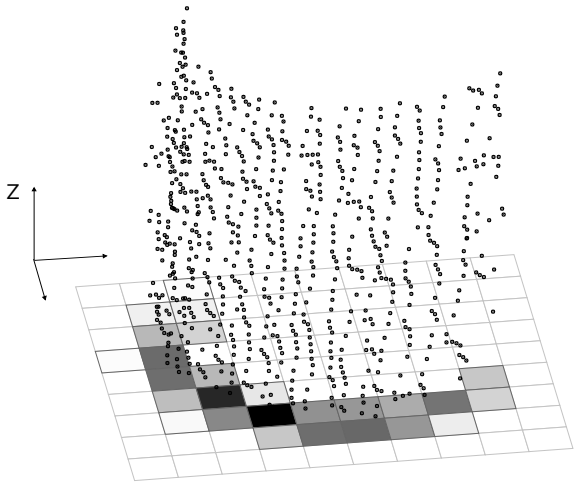


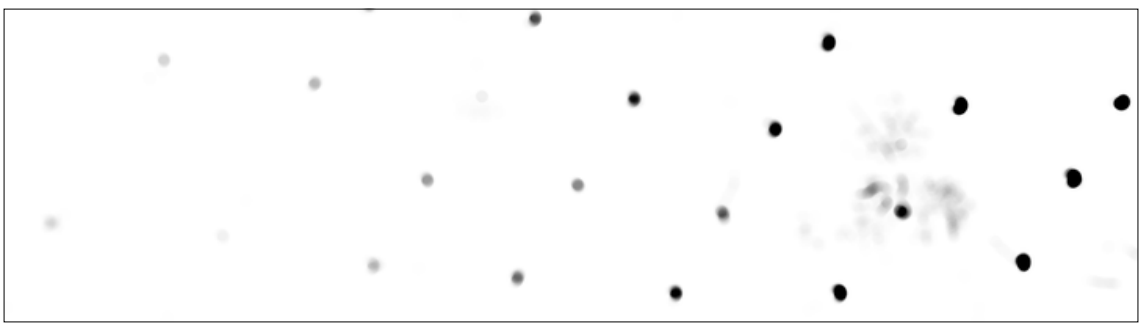
Figure 5.2: 3D points are projected onto a 2D grid. For points on trunks, a characteristic arc shape forms in the grid.

onto the same 2D grid cell, but incorporate the values of the 2D histogram of a slice directly. Thus, an accumulator unit is increased by the value of a source cell in the 2D histogram. As a result, peaks in the accumulator grid appear much more distinct, as indicated in figure 5.3. However, noise and other objects that have circular shaped cross-sections are amplified to some degree as well.

Center points of estimated circle- and arc-shaped point clusters emerge as peaks of high values in the accumulator grid. Though, the highest peak is not necessarily the center point of the trunk, which is sought. Often enough, trunks are occluded by other vegetation. As a result, a bin may contain only a fraction of the trunk surface that exhibits the characteristic arc-shape in cross-section. Due to other vegetation present in the same data set, it does not always cause the highest peak in the accumulator grid. Therefore, it is adequate to consider a number of local peaks in the accumulator grid.

(a) 3D points from a horizontal slice through data set E projected onto the XY plane.

(b) Resulting accumulation grid if CHT is applied to (a).



(c) Resulting accumulation grid if DHT is applied to (a).

Figure 5.3: Comparison of CHT and DHT. The grayscale colormap is the same for (b) and (c).

Hence, the accumulator grid is thresholded with parameter t_G. The resulting "islands" of local peaks are determined with CCL as explained in section 2.3. For each obtained connected component, the component cell with the highest value is identified as the targeted circle center point. Finally, the 3D coordinates of the considered grid cell are calculated. A circle is fitted to a set of points that are located within radius r around the obtained center point. If the fitted circle satisfies the requirements, i.e. if the circle radius is within a valid interval, the calculated center point is added to the set of candidate points for retrieval of the actual trunk centerline.

5.2.2 Recovery of the Trunk Centerline

The set of retrieved candidate points most likely contains a subset that can be interpreted as a point sequence of a polyline tracing the trunk centerline, as depicted in figure 5.4. Trees in data set B are growing rather straight. For this reason, the trunk centerline can be modeled by a straight 3D line segment. With the objective to find a subset of candidate points that are located along an unknown but discernible straight line, we employ RANSAC [Fischler-1981] with a 3D line segment as model template.

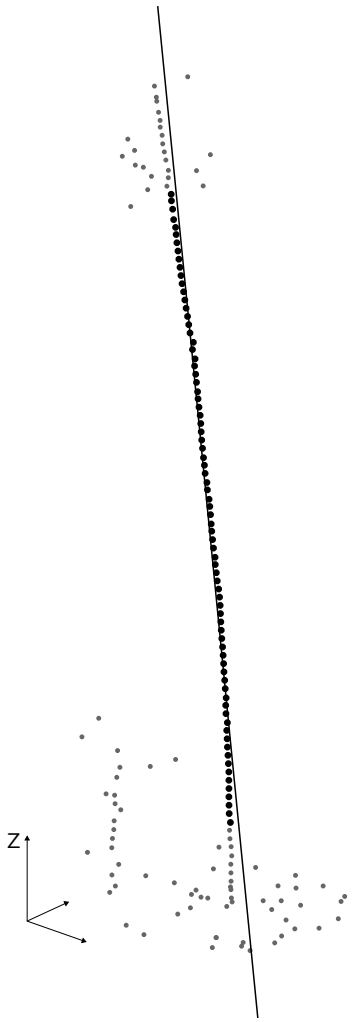


Figure 5.4: The set of inliers and the line segment that had been determined by RANSAC from the set of circle center points that were computed by applying the DHT to horizontal slices of the input data.

Since the number of candidate points is usually rather small, RANSAC quickly retrieves a suitable subset of inliers. Furthermore, the trunk centerline is often the only discernible line in the candidate point set. The set of inlier points is sorted along the Z axis. Naturally, each bin can only contain a single estimate point, which contributes to the trunk centerline. Subsequently, the retrieved inliers are successively connected to a polyline.

5.2.3 TLS Data Preprocessing in Voxel Space

We utilize a voxel space to retrieve the branching structure that is connected to the obtained trunk structure. A voxel space is a constant, rectilinear grid that partitions a region of continuous 3D space into cells. Usually, each cell is of cubic shape and called a voxel in analogy to a pixel. The extent $E = (E_x, E_y, E_z)$ of a voxel space $V \in [0 \dots E_x) \times [0 \dots E_y) \times [0 \dots E_z)$ that envelops the entire point cloud $P = \left\{p_i = (x_i, y_i, z_i) \in \mathbf{R}^3 | i = 1..K\right\}$ can be determined as

$$E = [E_x \; E_y \; E_z]^T = \left\lceil (\max P - \min P) \cdot \frac{1}{s_v} \right\rceil \tag{5.2}$$

where s_v is the voxel size, i.e. the length of a cell wall.

A point $p_i \in P$ is mapped to voxel space V with

$$p_i' = \left\lfloor (p_i - \min P) \cdot \frac{1}{s_v} \right\rfloor \tag{5.3}$$

As a result, a number of points in $\mathbf{R}^3$ may be mapped to the same voxel, which is governed by the voxel size. The voxel size is a crucial parameter, which controls the total extent of the voxel space. With decreasing voxel size, the voxel space exhibits an exponential growth behavior. Therefore, the employed hardware is usually the limiting factor for actual feasible processing unless this effect is specifically treated in implementation.

In addition to the number of points that map to a voxel, the centroid of this point group is associated with the voxel. Figure 5.5 shows a voxel space of a tree from data set B. In the following, we consider voxels to be occupied voxel cells. Voxels without any 3D points mapping to it or any other special property such as *trunk* are considered to be empty space.

The polyline that approximates the trunk centerline is mapped to voxel space, as well. Voxels, which are intersected by the trunk polyline, are tagged as *trunk*. Clearly, a significant number of trunk voxels are actually empty voxels because those particular voxels are located in the trunk interior, i.e. beneath the trunk surface, which was captured by TLS.

Furthermore, a high number of voxels that represent ground surface is present. Since no branching structure is expected at the tree foot, the voxel space is processed layer-wise and such voxels are cleared. Each voxel layer parallel to the XY-plane is tested separately until a predefined height is reached. In a layer, we locate the trunk voxel. Voxels that are outside an axis-aligned box, which is centered at the trunk voxel, are set to empty. The side-length of the box is determined manually in advance.

As a next step, we filter voxels by their number of points in order to carve out the tree surface that has been sampled densely. If the number of points of a voxel is below a predefined threshold N_p, the voxel is cleared. However, trunk voxels are a special case and remain unchanged. Subsequently, CCL as explained by Shapiro and Stockmann [Shapiro-2001] (see section 2.3) is applied to the voxel space. In this case, we consider adjacent voxels to be connected, if neither of the voxels is empty. Again, trunk voxels are handled as a special case and considered to be non-empty. As a result, each voxel is associated with a component label indicating to which connected component

Figure 5.5: A tree from data set B as (a) 3D point set and (b) corresponding voxel representation.

it contributes as illustrated in figure 5.6a. Subsequently, we assume that the main tree shape is represented by a comparably large connected component.

But before the considered connected component is isolated, two additional preprocessing steps are performed in order to preserve more details of the actual tree shape. First, we filter the voxel space for stray voxels. After CCL, a number of connected components that consist of single voxels can be identified. Such voxels are considered to be noise and eliminated.

Second, we join connected components that are in close proximity. If the voxel space is inspected, a number of connected components that apparently represent branch fragments are disconnected from the considered tree component as demonstrated in figure 5.6b. Therefore, we create artificial links to join connected components if the two disconnected components are only one empty voxel apart. We test every unoccupied voxel if its 26-adjacency contains voxels of different labels. In this case, we consider the unoccupied voxel as a linking voxel and treat it as a special case. All voxels of the connected components that are joined by the created link voxel obtain a new label indicating their unity.

Afterwards, all connected components but the considered tree component are eliminated from voxel space. Since trunk voxels are already tagged, we can identify this particular connected component as the largest one that has trunk voxels. Figure 5.7a shows the isolated tree component.

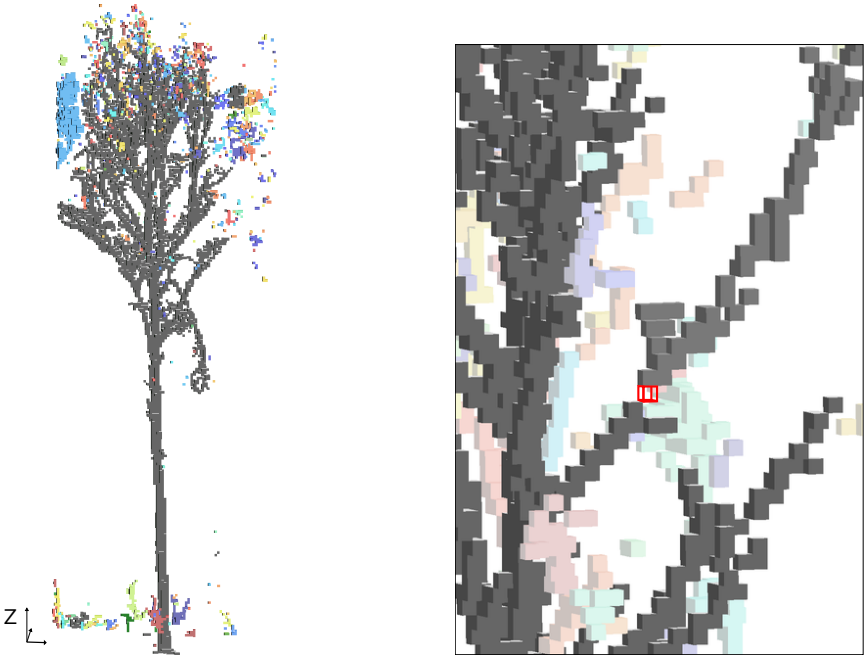


Figure 5.6: Voxel space representation after labeling by CCL. Labels are indicated by colors. (a) The gray component is the largest component and considered to be the sought tree. (b) Creation of a link voxel, indicated in red, between the tree component and an apparently disconnected branch.

All remaining voxels are part of the same connected component. If the voxel space is interpreted as a graph with voxels as vertices and adjacency relations as edges, a path that connects each pair of vertices can be found. Since our objective is to find paths from branch tips to the trunk, which is already known, a set of branch tips has to be established.

In the following, *apex* voxels denote voxels that supposedly represent branch tips. In order to select a set of apex voxels, each voxel is weighted by the minimal number of voxels that have to be traversed until the lowest trunk voxel is reached, which is summarized in algorithm 5.1. As illustrated in figure 5.7a, weights are monotonously increasing and reach local maxima at extremities. Subsequently, an apex voxel is identified as a voxel that has only equal or lower weights in its 26-adjacency. Clearly, only voxels that have an adjacency, which is not fully occupied, conform with this constraint. The resulting set of apex voxels is small in comparison to the total number of component voxels as depicted in figure 5.7a.

5.2.4 Retrieval of the Branching Structure by Searching

A set of trunk voxels and a set of apex voxels in voxel space were obtained. Hence, we propose to apply a search algorithm to find a path through the voxel component from an apex voxel to the trunk. As mentioned previously, each branch is eventually connected to the trunk. Therefore, we employ trunk voxels as goal states. A search is performed for each apex voxel to find a sequence of voxels that leads to a goal state.

The search starts at a voxel that is considered to be an initial state. At each state, possible next states are evaluated. In other words, the neighborhood of the current voxel is examined for a suitable candidate to move to. The search advances to the next state, i.e. the yet unvisited neighbor n, which minimizes the path cost function g. The path cost function must not overestimate the actual path

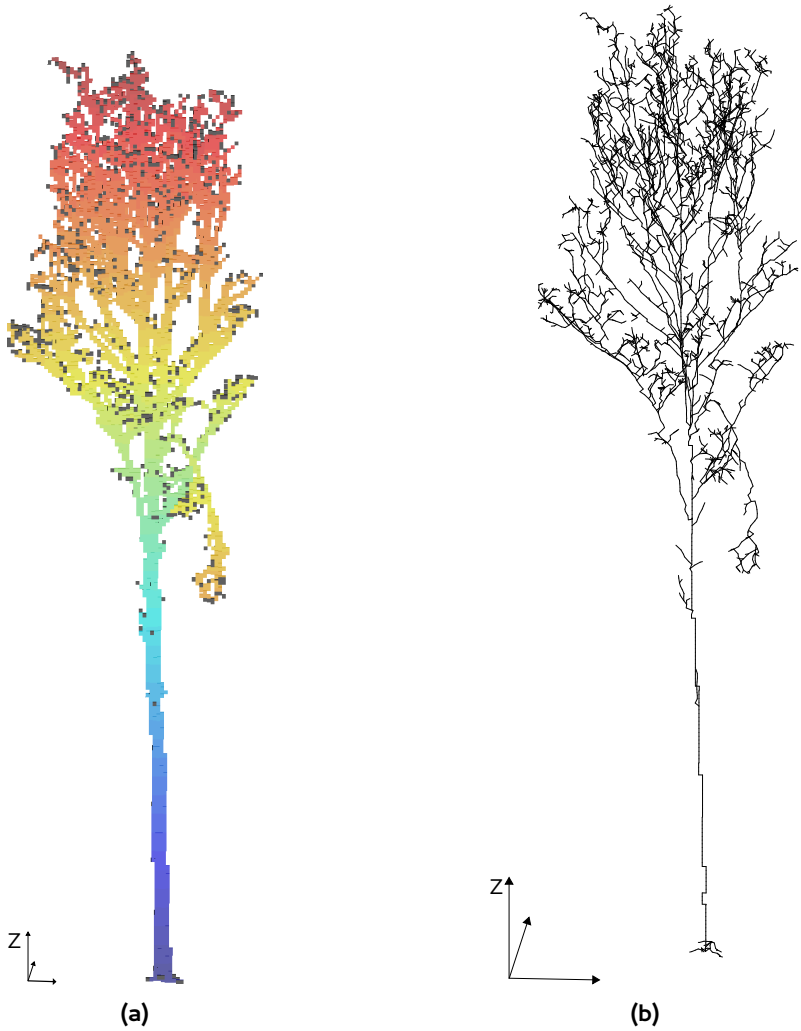


Figure 5.7: (a) Isolated tree component with a colormap indicating the smooth increase in weights starting at the bottom. Apex voxels are indicated in gray. (b) Resulting tree graph after searching a path from each apex voxel through the component to the trunk.

cost [Russell-2010]. Therefore, the previously assigned weights are utilized as g. In cases, where adjacent voxels have the same voxel weight, the Euclidean distance from the voxel to the closest trunk voxel is used for decision-making. In this way, the question which neighboring voxel should be visited next is solved by making the locally optimal choice. As a consequence, the performed search is an informed, greedy search.

In addition, testing for goal states or already traversed paths in the adjacency of a state gets priority over minimization of the path cost function. First, if a neighbor represents a goal state, it is immediately selected to move to. Second, if a voxel is discovered that has already been traversed by another path, the current traversal is terminated and both paths are linked. Again, the Euclidean distance to the closest goal state is employed to decide if there are more than one candidate voxels in the adjacency. As a result, searches are terminating faster the more paths have already been retrieved. If the current voxel has no goal states, no voxels with path indices nor unvisited voxels in its neighborhood, then the path is backtracked until unexplored voxels are encountered. In the worst case, the traversed path is traced back to the initial state. If a voxel is backtracked, the assigned path index of the current search is removed.

Each apex voxel is considered as an initial state and a search is started. The visited property of voxels is reset before the next search is performed, otherwise it may block subsequent searches. Since the path cost per voxel does not change between searches, the Euclidean distance of a voxel to the closest trunk voxel is computed in advance.

The result of a search is a sequence of voxels, which is added to a graph T that represents the spatial tree structure. For each voxel that is not yet included in T, a vertex is added to the graph. The centroid of 3D points that map to the particular voxel, which was determined earlier, is utilized as 3D coordinates of the new vertex. If the voxel is a special case such as a *link* or *trunk*, the voxel center point is employed as vertex coordinates and transformed to the coordinate system of the 3D source data accordingly. Between each pair of subsequent voxels in the sequence, an edge connecting the corresponding vertices is added to graph T. Figure 5.7b shows the resulting tree graph T that was obtained from the voxel space in figure 5.5.

After assembly of the tree graph, we perform a relabeling of vertices summarized in algorithm 5.2, which allows sorting all paths by size afterwards. As a result, information about the branch a vertex belongs to and the branch length can be easily retrieved from the tree graph. Furthermore, the longest path then contains all trunk vertices.

5.3 Experiment Setup and Results

We have conducted experiments with the 42 selected Silver Birch trees from data set B (see section 4.1). In figure 5.8 and 5.9, selected experiments results are shown. For each tree, a bounding box of cylindrical shape was cut out from the multiscan. The Z axis at the given tree position is considered to be the main axis of the bounding cylinder. The radius of the cylinder was set to 4.3 m . The average number of 3D points within the chosen bounding cylinders is between 388,198 to 7,073,548 points.

The DHT was implemented in Mathworks MATLAB R2009a (32 bit). Only 3D points below a height of $Z = 13.0$ m were considered for the procedure in order to skip processing of points that most likely belong to the crown. Slice thickness was set to $t_Z = 0.15$ m. A cell size of $t_c = 0.01$ m was used to discretize the 3D points to the 2D raster. Similarly, the disc radius was set to $r = 0.15$ m. After preliminary testing, the thresholding parameter was defined as $t_G = \max A \cdot \frac{1}{14}$ points in a raster element. Circle fitting is performed using a MATLAB script [Chernov-09b] that implements the circle fit method of Pratt [Pratt-1987]. Circles need to have a radius larger than 0.07 m to be considered further; otherwise the circle is discarded. The threshold was included on the basis that trees in data set B are unlikely to be thinner than 0.14 m in diameter.

The set of points located within a cylinder of radius $r_c = 1.0$ m at the given tree position is considered further. A straight line segment is retrieved from this point set with RANSAC with a 3D line model using a MATLAB script by Kovesi [Kovesi-00]. Overall processing in MATLAB takes about 15.9 to 90 seconds for each of the 42 tree data sets. Clearly, processing time is chiefly depending on the number of points, i.e. the bounding volume that is considered.

The intermediate result was further processed by a C++ implementation utilizing the Eigen [Eigen] and Boost [Boost] libraries on a dual-core machine (4 GB RAM, Linux, 64 bit). The voxel size was set to $v_s = 0.1$ m for the voxel space analysis. A ray tracing algorithm [Williams-2005] was used in order to trace the obtained trunk polyline through the voxel space and tag voxels accordingly. The minimum number of projected points per voxel was set to $K = 10$. Iterative searching is rather fast with 1.5 s to 54.3 s for 161 to 5001 paths, respectively. Obviously, memory consumption is governed by the voxel size setting. In general, the implementation is not particularly optimized. On average, generation of a tree graph takes about 37.85 s time (min. 12.7 s and max. 105.24 s).

Algorithm 5.1 Weighting scheme

```
1  initizalize stack Q ← ∅
2  for all voxels v in component C do
3      weight(v) ← ∞                          ▷ A weight of ∞ signifies an unweighted voxel
4  end for
5  weight(v_root) ← 0                ▷ Assign zero to lowest trunk voxel v_root of component C
6  Q.push(v_root)
7  while Q ≠ ∅ do
8      v_cur ← Q.pop()
9      N ←neighbors(v_cur)                          ▷ Retrieve set of neighboring voxels.
10     if weight(v_cur) = ∞ then
11         w_min ← minimum weight of all n ∈ N
12         weight(v_cur)← w_min + 1
13     end if
14     for all n ∈ N do
15         if n ∉ Q then
16             Q ← Q ∪ n
17         end if
18     end for
19 end while
```

Algorithm 5.2 Relabeling

```
1  procedure RELABELVERTICES(G: tree graph )
2      set S ← ∅
3      for all vertices v ∈ G do
4          weight(v)← −1                          ▷ Initizalize weights of all vertices
5          if degree(v) = 1 then
6              S ← S ∪ v                          ▷ Select all graph vertices of degree one
7          end if
8      end for
9      i ← 0
10     for all vertices v in S do
11         i ← i + 1
12         v_cur ← v
13         v_parent ← getParent(v_cur)          ▷ Except the root, every vertex has a parent
14         id(v_cur) ← i                                ▷ Each path obtains an index
15         weight(v_cur) ← 0
16         while v_parent ≠ ∅ and weight(v_parent) ≤weight(v_cur) do
17             weight(v_parent) ← weight(v_cur) + 1
18             id(v_parent) ← id(v_cur)
19             v_cur ← v_parent
20             v_parent ← getParent(v_cur)
21         end while
22     end for
23 end procedure
```

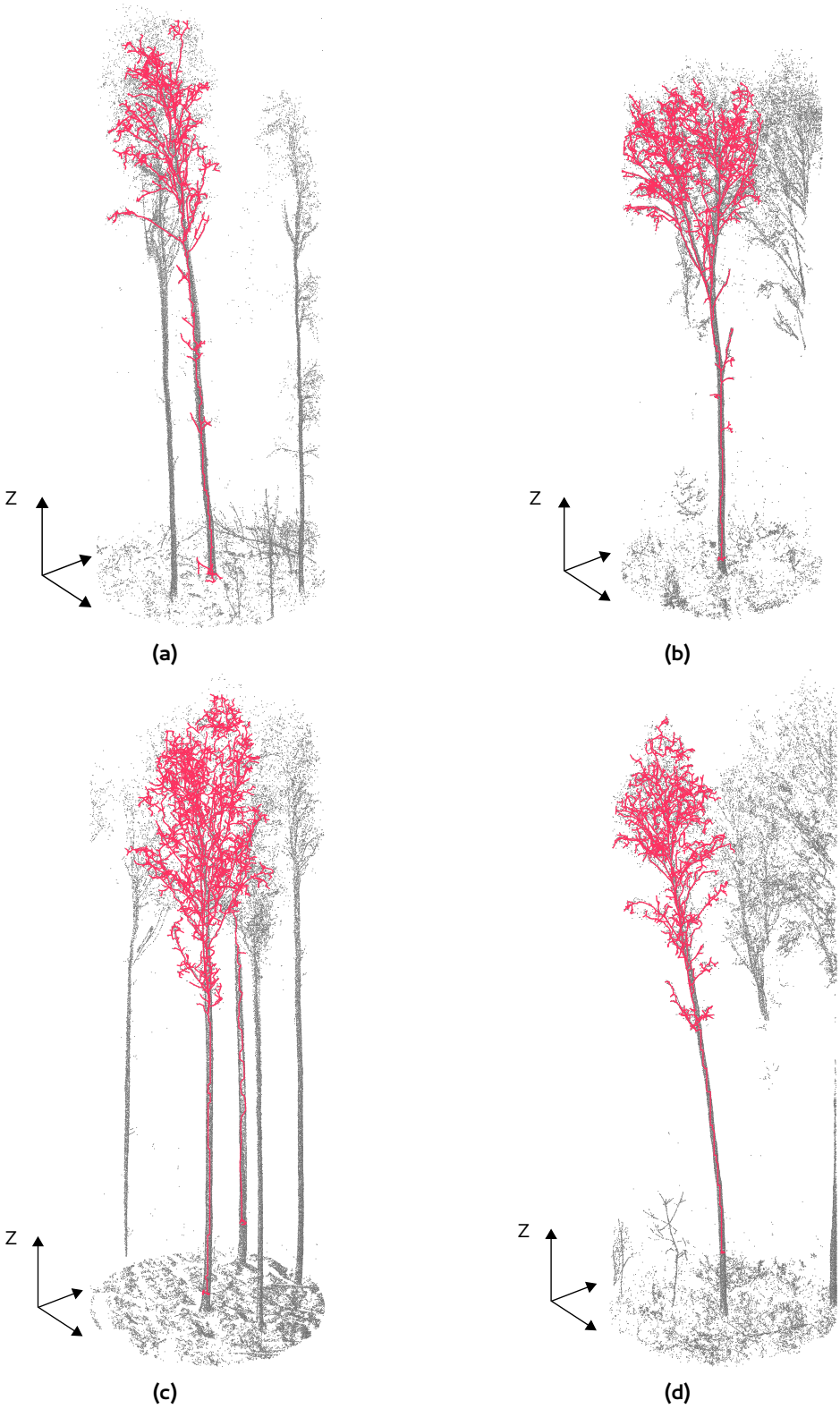


Figure 5.8: Results of the experiments. The retrieved skeleton graph is drawn in red as overlay onto the 3D point set in gray. Subfigure (c) shows a case where the neighboring tree trunk is included in the skeleton due to intertwined crowns.

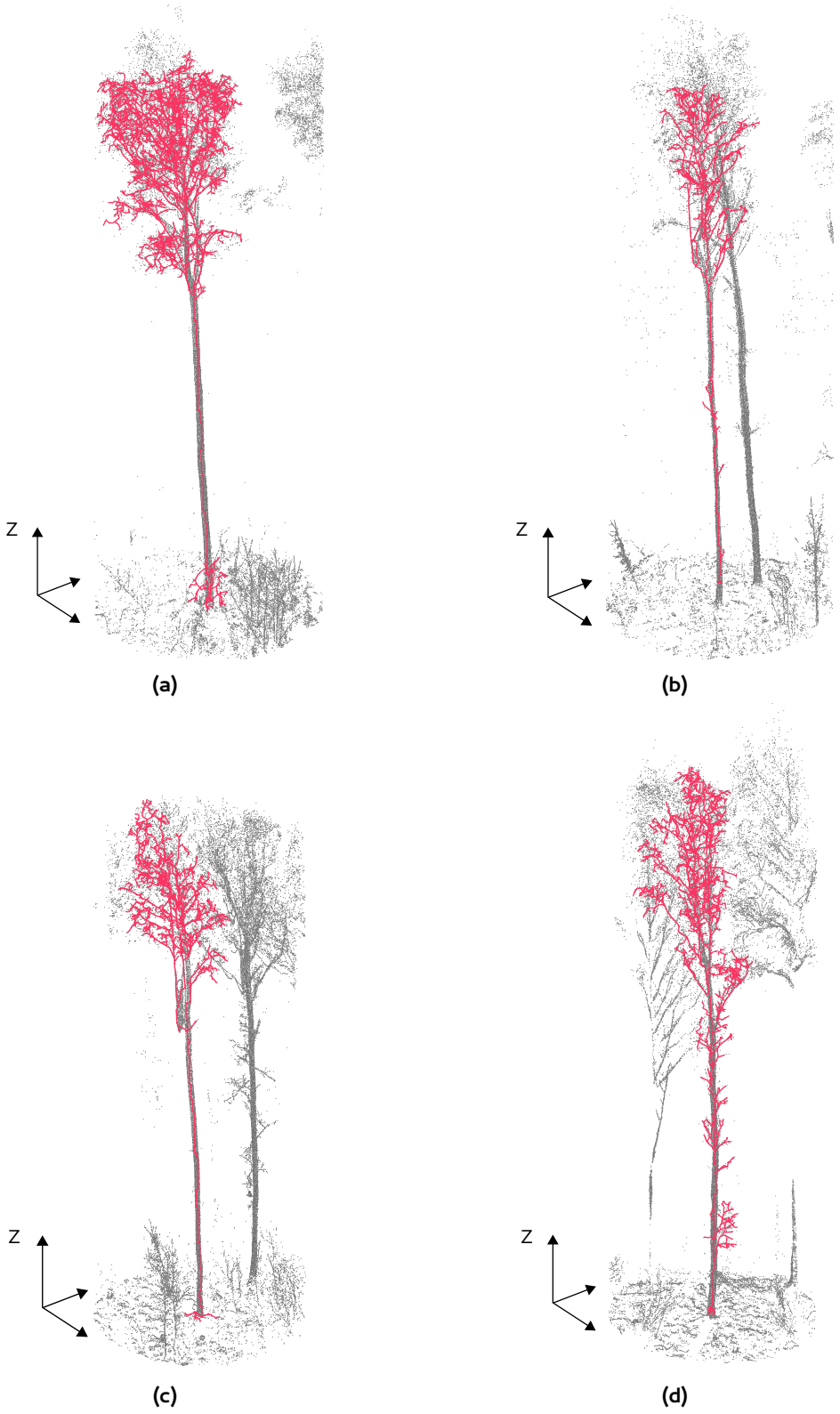


Figure 5.9: Results of the experiments. The retrieved skeleton graph is drawn in red as overlay onto the 3D point set in gray. In subfigure (c), apparently strange artifacts are visible at the beginning of the crown that were caused by wind during a scan.

5.4 Discussion

The experiments have shown that the presented method provides an adequate solution to rapidly generate a tree graph representing the spatial structure of the scanned tree. Processing times of the experiments can be considered to be an indicator that a non-optimized implementation completes the task in a satisfying amount of time. Consequently, there are possibilities to enhance the performance of the procedure. In addition, the overall memory requirements are low. They depend mainly on the chosen voxel size, which controls the extent of the voxel space. In this way, efficient processing is achieved. At the same time, the level of detail of the tree graph and the degree to which it represents the perceptible tree in the TLS data set is limited.

In the following, the subprocedures are assessed separately: First, the DHT seems to be an adequate modification for TLS data. In contrast to CHT, the point density of the 3D point cloud is directly exploited, which was pointed out by Schilling et al. [Schilling-2011a]. Structures with near perfect circular cross-sections that are sampled by a high number of points result in distinct peaks in the accumulator grid. Tree trunks face the scanner with the largest patch of continuous surface in comparison to branches or understory vegetation. Consequently, trunks are densely covered with points, which cause the highest peaks in the accumulator grid. Of course, the actual scan coverage and representation quality depends on the distance of the object to the scanner, as well as occlusions. But as the experiments show, the procedure is well suited to detect cross-sections of tree trunks in practice.

The parameter t_G that is used to threshold the accumulator grid chiefly controls the result of the procedure. A low threshold results in more peaks that are evaluated. If the threshold is too high, trunks that are not as densely sampled in contrast to nearby understory vegetation may be removed. The threshold parameter was defined in dependence of the accumulator element of maximal value after empirical testing. Clearly, the current strategy to filter the accumulator by a global threshold is a working, but suboptimal solution. It would be beneficial to test alternative strategies to improve results.

The precision that is lost due to the disc template is recovered by circle fitting. Schilling et al. [Schilling-2012b] employed a circle fit by Kasa [Kasa-1976; Chernov-09a]. However, this algebraic fit is biased when the points cover only a small arc. For this reason, we use the circle fitting method that was proposed by Pratt [Pratt-1987].

Furthermore, we have conducted preliminary tests with a ring template instead of a disc in the accumulation process. The objective was to test whether a ring is a suitable template to compensate slight point displacement and enforce a minimal radius of the circular structure at the same time. However, the results have shown that a ring has no clear advantage over the original CHT formulation. The peaks in the accumulator grid are not as distinct as with the disc.

The retrieval of a set of circle centers by DHT is controlled by few parameters. The parameter values that were used in the experiments were set after empirical testing with data set B. Similar to the voxel size, the slice thickness and the cell size have great impact on the processing results. Slice-wise processing was limited to a certain height in order to omit crown points, since the branching structure is targeted in the next computation step. The objective of the slice-wise application of the DHT is to obtain a sufficiently large set of candidate points to determine a line segment with RANSAC. We do not aim to model the entire trunk length in this subprocedure.

Moreover, we only consider center points of circles that have a radius larger than a minimum value. Valid trunk centers that are located at higher heights may be lost with this radius-based filter due to the taper of the trunk, i.e. decreasing diameter of trunk with height. This can be neglected because the objective is to retrieve a sufficiently large part of the trunk centerline to tag

as trunk. In the experiments, this is always achieved. However, if the processing aim is altered then parameters and subprocedures have to be adapted to the new task.

The trunk centerline is quickly retrieved by RANSAC because the set of circle centers is comparably small. For a larger set of candidates, RANSAC may perform considerably more iterations in order to find a best-fit solution. Beside the actual content of the slices, the parameters that are used for peak detection have a significant influence on the cardinality of the candidate set and consequently also on the performance of RANSAC. A 3D straight line segment is utilized as model by RANSAC. For the data set B, the assumption that trunks can be represented by a line segment is reasonable because the Silver Birches grow rather straight. Even if they have a slightly bending trunk, a limited set of trunk center points that confirms with the model can be found as the experiments have shown.

After retrieval of the approximate trunk centerline, further processing is performed in voxel space. The voxel size is of essential importance. If the voxel size is large, less memory space is needed for the voxel space, which is then a rather down-sampled representation of the source data. If the voxel size is small, more memory is consumed and the voxel space represents the data with a higher level of detail. At the same time also data gaps are directly reflected in voxel space as empty voxels causing a considerably fragmented representation of the data. Hence, a finer resolution due to a small voxel size does not necessarily result in a more complete and detailed tree graph. On the contrary, a lot more tree parts may be missing because of the resulting higher number of connected components, as demonstrated in figure 5.10. For the experiments, voxel size was set to $v_s = 0.1$ m. By empirical testing, we found that it is an adequate setting to achieve a satisfying balance between a sufficiently detailed voxel representation, maintainable memory requirements, and observable completeness of the results. The defined voxel size is a reasonable choice that complies with the demanded spatial resolution w.r.t. the utilization of the retrieved structural description in the context of a radiation transfer model, which was the original intention during development.

After mapping the 3D point cloud to voxel space, trunk voxels are marked. The line segment that is spanned by two adjacent polyline vertices is traced. Voxels that are intersected are tagged as trunk. Tracing of line segments is performed using a ray tracing algorithm [Williams-2005]. It may seem overly elaborate to apply a ray tracing algorithm for this task, though it is certainly necessary to ensure a gapless chain of trunk voxels. Furthermore, the algorithm by Williams et al. [Williams-2005] specifically targets ray tracing though a 3D grid and solves the task efficiently.

Closely related to the voxel size is the filter size N_p for the removal of voxels that are only sparsely occupied by 3D points. In fact, data set B represents trees in leaf-less state, but when inspected visually, a high number of apparently noise points are present in tree crowns. As a result, elimination of voxels that have a low number of points is crucial to uncover the relevant woody tree parts which are usually sampled more densely. Obviously, if the value is too large, a significant fraction of proper voxels are eliminated as well. If the value is too small, filtering is ineffective. The filter size was set to $N_p = 10$ for data set B, which achieves a seemingly reasonable filtering by visual examination.

Filtering is the crucial factor that influences how many connected components are determined by the following CCL procedure. If the number of connected components is high, it is likely that more parts of the tree are eventually missing from the tree graph because they are obviously represented by disjoint components. If the number of components is very low, the resulting tree graph might be a too coarse description of the real-world tree. However, the voxel size setting has to be taken into account to assess the situation.

Since filtering is required, creation of link voxels between components that are only disjoint due to one single (emptied) voxel seems like an adequate compromise to moderate the effect of filtering. The subprocedure is based on visual examination of the voxel space after filtering and CCL. We

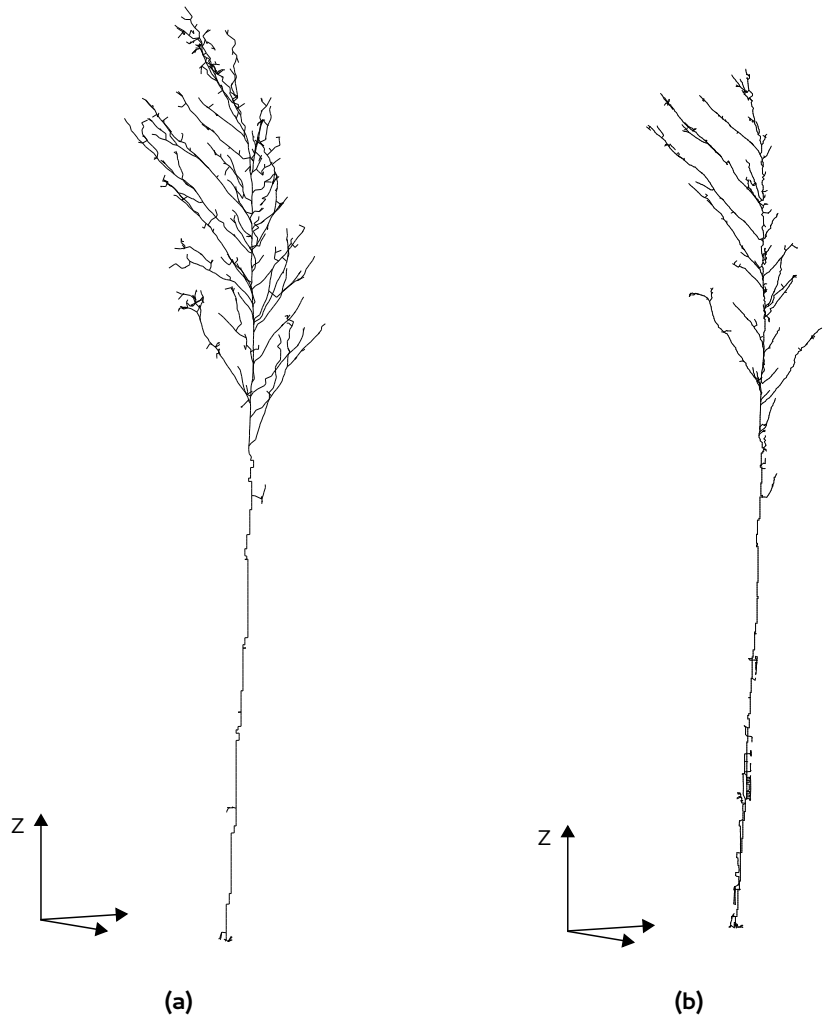


Figure 5.10: Tree graph as result of a voxel space representation with (a) a voxel size of 10 cm and (b) a voxel size of 5 cm.

found that the positive effect of re-attaching a strong branch to the chief tree component predominates the possibly negative influence of appending small, insignificant components. Naturally, connections between disjoint components that are separated by more than a single voxel are not created. As a consequence, some tree parts are definitely missing in the final result if the source data has relevant data gaps that are reflected in voxel space. Without at least a reasonable hypothesis about the component shape w.r.t. its semantic meaning for the tree and its direction of growth, it seems counter-productive to connect those components. However, an analysis of the voxel space that considers different levels of resolution as done by Gorte [Gorte-2006] may provide a solution to this problem.

Presently, weights, which are assigned to voxels in order to identify apex voxels, coincide with the number of steps required to reach a voxel with a Breadth-First traversal starting at the lowest trunk voxel. The weighting scheme is closely related to a distance transform. Most often, the determined apex voxels truly represent extremities of branches. However, a number of apex voxels, which are identified with the selection criterion, are not considered appropriate candidates on visual judgment but rather represent small perturbations on the tree surface as shown in figure 5.7a. Consequently, an equal number of spurious paths are created in the tree graph. A revision of the strategy to find apex voxel with the objective to decrease the number of false positives would be

beneficial. Distance transforms with various metrics could be tested whether they achieve better results, for instance. Though, the detection of apex voxels cannot be completely solved if only the 26-adjacency of a particular voxel is considered. In general, a larger region of interest needs to be examined.

The core mechanism of the approach to retrieve the branching structure is the informed greedy search. A consequence of the greedy heuristic is that the resulting search paths are not necessarily globally optimal paths. Instead, paths are the result of taking the locally optimal decision at each state with regard to the given path cost function. For a plausible reconstruction, it is important that the path cost function appropriately describes the distance of a voxel to the goal state through the interior of the connected component. We found that the Euclidean distance from each voxel to the closest goal state does not produce satisfying results, for instance, if voxels resemble a hollow tube in the upper trunk section, but an incomplete tube in the lower trunk section. In this case, a search might follow the voxels down the tube side that is not connected to the lowest trunk voxel because the path cost function indicates that a goal state would be spatially near. Only after exploring all adjacent voxels, the search actually backtracks to the point where a path to the goal state can be found at all. Therefore, the weights that were assigned for the detection of apex voxels are utilized in the path cost function. The Euclidean distance is only used to decide, from which of the equally weighted neighbors the goal state might be quicker to reach in contrast to picking one of them arbitrarily.

Presently, the tree graph structure is not globally centered in the given 3D point cloud. If a path includes a voxel that provides a centroid of the 3D points, which were mapped to it, the vertex is centered in this small point cluster. If a path traversed a trunk or link voxel, which are unoccupied, the coordinates of the voxel center are utilized. As a result, the tree graph is not always smooth. [Gorte-2006] propagates labels of the skeleton located in the interior of the voxel component towards the exterior of it. Obviously, the application of a similar strategy with the objective to improve the centering of the retrieved skeleton on the basis of 3D points within voxels of the same label would be beneficial.

Given these points, the resulting tree graph is an adequate description of the spatial structure that is contained in the data set. The main branching structure is retrieved rapidly and reliably with respect to the provided scan coverage. The quality of the fine branches and twigs is suboptimal, though causes and possibilities to improve it were already pointed out. All in all, the obtained skeleton graph can be considered to be a fast approximation of the real-world tree.

It is important to realize that the input data sets are cutouts of a larger multiscan representing an entire study site, in contrast to the majority of data sets in the studies that were discussed in section 2.3. Unlike these setups in which a particular tree is the single, central object of a multiscan, the tree may be represented with poor detail in our source data. In other words, the scan coverage of the tree in the data sets is not checked in advance. Anyway, we ensured that the 42 trees of data set B that were selected for the experiments had been scanned from more than one viewpoint. Clearly, data from a single viewpoint is insufficient if a complete skeleton is sought. However, the definition of a bounding volume for cutting out a section of data from a multiscan is problematic. For the experiments, the bounding volume was a cylinder with the main axis parallel to the Z axis located at the given tree position. The choice of the cylinder radius is a difficult issue. If the radius is selected to small, the bounding cylinder will be too narrow for the tree and the crown will be inadvertently pruned. In addition, it has to be taken into account that the tree may exhibit a skewed growth, which is not reflected by the cylinder. If the cylinder radius is large, nearby trees may be included in the bounding volume. If there is a visible gap between tree crowns that is also reflected in voxel space, the presence of other vegetation within the bounding cylinder of the considered tree does not influence the processing. However, crowns of nearby trees often grow seemingly into one another. As a consequence, both trees emerge in voxel space as a single connected component. In this case, the longest path often connects the lowest voxels of both tree

trunks. At least, this could be employed as an indicator to recognize these cases. At the moment, such configurations are not addressed by the method.

Furthermore, the co-registration accuracy of the multiscan might be unsatisfying in some regions due to shifts in point locations that were caused by wind during the scanning. Since such effects are not compensated, a single branch may appear as two branches in the point cloud. Then, the resulting tree graph also includes two spurious branches instead of the true one. Point offsets caused by wind are a common problem to TLS processing. At present, there does not exist an established strategy to overcome those effects.

All things considered, the presented method relies on a few parameters that chiefly control the outcome. Although these parameters were fixed in all conducted experiments, they depend on the characteristics of the input data. Most likely, they need to be changed if other data sets are processed, which limits the general applicability of the method to some degree. As pointed out previously, development was mainly data-driven. Therefore, it is certainly necessary to test the method with other data to limit the effects of over-tuning the processing to a particular data set. In the scope of the thesis, this was not possible: Scots Pine as a coniferous species is not an appropriate input to the method, and Eucalyptus trees do not have enough branching structure to serve as a relevant comparison.

5.5 Future Work

Apart from the aforementioned improvements, there are a few aspects that would profit from revision. For instance, tree detection has not been addressed. Instead it was assumed that the position of the tree is known in advance, possibly as the result of the previously mentioned tree detection methods or by manual selection. Integration of this task into the presented method would be beneficial.

In fact, the method is in principle not limited to receive single tree data sets as input. If an appropriate data management strategy is implemented, such that the multiscan data is available at runtime, the DHT could be applied to a larger region of interest. Consequently, several trunk centerlines could be obtained in the same processing step. Moreover, RANSAC is not the only way to retrieve them. If the tree trunk was sufficiently sampled by 3D points, the DHT will detect a trunk circle in each slice. A tree commonly grows rather monotonously upwards. For this reason, we can expect that the displacement between centers of trunk circles of two adjacent slices will be limited with respect to the slice thickness. As a result, RANSAC could be replaced with a tracking scheme in order to simultaneously track several trunk centerlines during application of the DHT to each separate slice.

Furthermore, utilizing an Octree or a kD-tree instead of a voxel space with a constant grid may be advantageous.

6 COMPONENT BOUNDARY TRACING IN TLS SCANS

In this chapter we introduce a novel approach to handle scan data from TLS. On this basis, we present a strategy to trace the boundary of components that have been determined by Connected Component Labeling [Shapiro-2001] on range data. The boundary tracing is inspired by a graph-theoretical approach [Gross-1987], which enables to include depth discontinuities of a component. Finally, we present experiment results and discuss the proposed strategy in detail, which lays the foundation for the skeleton retrieval of the following chapter.

6.1 Motivation

TLS data is usually interpreted as a set of 3D points. Since explicit adjacency relations are not provided, it is understood as being unstructured data. The lack of inherent neighborhood relations in 3D point clouds is therefore the main argument for the voxel space approach. However, as discussed in the previous chapter, the voxel size has great influence on the overall processing performance, which makes it a critical task to define it appropriately.

Although a 3D point cloud is often considered unorganized data, it is also obvious that a single scan contains point neighborhood information due to vertical and horizontal laser beam deflection in fixed increments $\Delta\phi, \Delta\theta$. As a matter of fact, TLS data of a single scan is highly ordered data due to the sequential scanning process that is summarized in algorithm 6.1. Clearly, preserving the original scan sequence as raster is beneficial for processing.

Algorithm 6.1 Scanning process

```
1  N ← ⌈φ_max/Δφ⌉                          ▷ Number of increments in horizontal plane
2  M ← ⌈θ_max/Δθ⌉                          ▷ Number of increments in vertical plane
3  for x ← 0 ... N − 1 do
4      φ_x ← x · Δφ
5      for y ← 0 ... M − 1 do
6          θ_y ← y · Δθ
7          q ← x + y · N                   ▷ Index of measurement
8          perfom measurment at (φ_x, θ_y)
9          determine intensity w, and range r
10         compute 3D Cartesian coordinates p_q ← (X_q, Y_q, Z_q) from (φ_x, θ_y, r)
11     end for
12 end for
```

Information about the scan sequence is often available in form of the PTX format from TLS software. Since PTX is a plain text format, we have developed a binary data container in order to facilitate prototyping with TLS data in its original raster alignment. Accessing TLS data as a raster alignment allows the application of image processing methods because a 3D point can be queried like an image pixel. Consequently, we propose to identify point groups that are supposedly sampled from the same object surface via CCL. Subsequently, we present a method to retrieve boundary information from connected components, which is able to trace inner depth discontinuities as well.

6.2 TLS Data in Raster Alignment

Most often, point clouds are exported as plain text files that can contain 3D coordinates, intensity, range, and the actual measured spherical coordinates. The exported data normally comprises only the measurements that yielded valid results. In contrast, laser beams that were emitted towards the sky are inevitably invalid measurements because no range can possibly be determined. Naturally, 3D coordinates cannot be computed if the range cannot be properly measured.

6.2.1 Intensity and Range Images

In order to generate a range or intensity image from the subset of valid measurements, the raster alignment is usually restored from the spherical coordinates ϕ, θ [Vosselman-2010; Eysn-2013; Bienert-2013]. If the increments $\Delta\phi$ and $\Delta\theta$ are known, the 2D image coordinate can be computed as

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \lfloor(\phi - \phi_{min})/\Delta\phi\rfloor \\ \lfloor(\theta - \theta_{min})/\Delta\theta\rfloor \end{pmatrix} \tag{6.1}$$

The resulting raster has the size

$$\begin{pmatrix} M \\ N \end{pmatrix} = \begin{pmatrix} \lceil(\phi_{max} - \phi_{min})/\Delta\phi\rceil \\ \lceil(\theta_{max} - \theta_{min})/\Delta\theta\rceil \end{pmatrix} \tag{6.2}$$

The laser deflection unit is a mechanical device. Therefore, the actual spherical coordinates exhibit slight offsets from the perfect grid-like spacing. As a result, more than one point may be mapped to a single element of the raster. Since a pixel cannot be subdivided further, a strategy for computing a single intensity or range value from multiple entries per raster element has to be chosen. Furthermore, a raster element may stay empty, which results in a data gap and thus a background pixel in the generated image. The need for interpolation and resulting data gaps are clearly a drawback when the raster is restored on the basis of noisy spherical coordinates.

According to [Vosselman-2010], the intensity values are scaled to the interval $[0 \dots 1]$ or $[0 \dots 255]$ to obtain a valid gray level representation. Often the intensity image is directly available as image export, for example as TIFF. Similarly, range values are used for mapping if a range image is the target output. However, the interval of range values is usually color-mapped to exploit the RGB color space. A limit to the 1D integer interval $[0 \dots 255]$ would result in a down-sampling of the range information.

Figure 4.1 shows an intensity image from data set B. Clearly, a scan as raster looks distorted similar to a Mercator projection of the earth globe. The farther rows are away from the row that corresponds to the scanner's XY-plane in SOCS, the more distorted object shapes appear in the raster. This effect is an inherent property of the mapping of a curvilinear spherical coordinate system onto a constant, rectilinear grid. Still, the raster alignment gives an overview of the entire scan and provides rather intuitive access to the TLS data.

6.2.2 The PTX format

A grid-like scan representation of single scans is often readily available from TLS software in form of the PTX file format. This export option is offered for instance by Z+F LaserControl [ZF-2012] and Trimble SCENE [Trimble-2013]. The scan order is not lost even if several scans are co-registered. Co-registration only affects the transformation matrix that transforms points from SOCS to PCS.

Admittedly, dependence on a particular export format is arguable. But the Imager series by Z+F as well as the Faro Focus 3D are popular instruments. Therefore, we considered it appropriate to investigate the beneficial properties of the PTX format.

PTX is a proprietary plain text format originally created by Leica Geosystems for the Cyclone software suite. A brief description of the format is available from the Leica Geosystems Knowledge Base [Leica-2012] and an example is presented in table 6.1.

20238	// nb of columns in raster
10000	// nb of lines in raster
27.627 -5.215 0.100	// scanner position
-0.509305 -0.860586 0.000126	// X axis of SOCS in PCS
0.860586 -0.509305 -0.000683	// Y axis of SOCS in PCS
0.000653 -0.000239 1	// Z axis of SOCS in PCS
-0.509305 -0.860586 0.000126 0.0	// 4x4 homogeneous transformation matrix SOCS $\rightarrow$ PCS
0.860586 -0.509305 -0.000683 0.0	
0.000653 -0.000239 1 0.0	
27.627 -5.215 0.100 1.0	
0.0000 0.0032 22.3854 0.000	// scan data
0.0000 0.0032 22.3859 0.000	// 3D coordinates as 3 float, intensity as 1 float
0.0000 0.0032 22.3746 0.000	// optionally RGB $\in [0 \dots 255]^3$ per line
0.0000 0.0032 22.3535 0.000	
0.0000 0.0032 22.3862 0.000	
0.0000 0.0032 22.3760 0.000	
0.0000 0.0032 22.3879 0.000	
0.0000 0.0032 22.3863 0.000	
.0 .0 .0 .0	// invalid measurement denoted by zeros
0.0000 0.0032 22.3796 0.000	
0.0000 0.0021 14.4961 0.000	
...	// listing reduced

Table 6.1: Example of a scan export in PTX format.

The first two integers denote the number of columns N and rows M of the raster A that is to be filled. The following four lines include translation and rotation of the SOCS to the PCS if a

co-registration was performed. The next four lines are occupied by a 4×4 homogeneous transformation matrix H that stores the same information. If the PCS coincides with the SOCS, the transformation matrix is identical to the identity matrix. The remaining $N \times M$ lines contain the actual point cloud data. Each line lists 3D coordinates of a point $p_q \in \mathbf{R}^3$, the intensity value scaled to $w_q \in [0 \dots 1]$, as well as RGB color values $rgb_q \in [0..255]^3$ if available.

The data tuple of a line corresponds exactly to one raster element. Its position in the raster can be calculated with

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} q \bmod N \\ \lfloor q/M \rfloor \end{pmatrix} \tag{6.3}$$

with $q \in [0 \dots (N \cdot M) - 1]$ as the index of the particular line, i.e. the number of lines that were encountered before. In other words, each subsequent set of N data tuples is interpreted as a row of the raster, which is built in row-major order from the given data set. The data set includes all valid measurements as well as all invalid ones. Invalid measurements are indicated by a line containing only zeros as in line 19 of table 6.1.

If the PCS is not identical to the SOCS, each point $p_q = (X_q, Y_q, Z_q)$ has to be transformed to PCS by the provided homogeneous transformation matrix $H_{4\times4}$ with

$$\tilde{p}_q = H^T \cdot \begin{pmatrix} X_q & Y_q & Z_q & 1 \end{pmatrix}^T \tag{6.4}$$

In general, it is unclear, which operations are executed in proprietary software between scanning and data export unless explained in the software documentation, which is hardly the case. The Z+F Imager 5006i takes a full scan as follows: A scan line is caused by deflecting the laser beam overhead in the vertical plane by a rotating mirror. The total amount of rotation of the scanner head in the horizontal plane is significantly smaller. With this in mind, the expected data raster would look like in figure 6.1a, whereas the actual PTX content appears to be the raster in figure 6.1c. Moreover, PTX export of Trimble SCENE provides the data raster as in figure 6.1b. The side-by-side arrangement of two scan parts results in a noticeable seam in figure 6.1c. At this seam, raster elements are not truly neighbors and would compromise any processing based on adjacency information in the raster. For this reason, we simply split the data set in two sides A and B if the PTX file includes such a seam. Trimble SCENE outputs are split accordingly regarding their horizontal seam. As a positive side effect, a data set is easier to handle if split in half.

To sum up, it is helpful to think of the recovered raster as a multidimensional image with stacked data layers. The available data items per raster element $A(x, y)$ from a PTX file are

- its 1D linear raster index q
- its 2D raster coordinates (x, y)
- 3D coordinates $p_q = (X_q, Y_q, Z_q)$ in SOCS
- 3D coordinates in PCS $\tilde{p}_q$ if different from SOCS
- intensity value w_q
- optional RGB values if imagery was available

In addition, the distance to the laser scanner can be restored from 3D coordinates as

$$r_q = \|\tilde{p_q} - p_{scanner}\| \tag{6.5}$$

with $\|\cdot\|$ denoting the Euclidean distance.

According to [Leica-2012], the PTX format was specifically created to allow other software the estimation of normals from scan data, which is facilitated if the grid-like scan structure is preserved. As a consequence, cumbersome restoration of the intensity or range image can be omitted if this format is available as export option in TLS software.

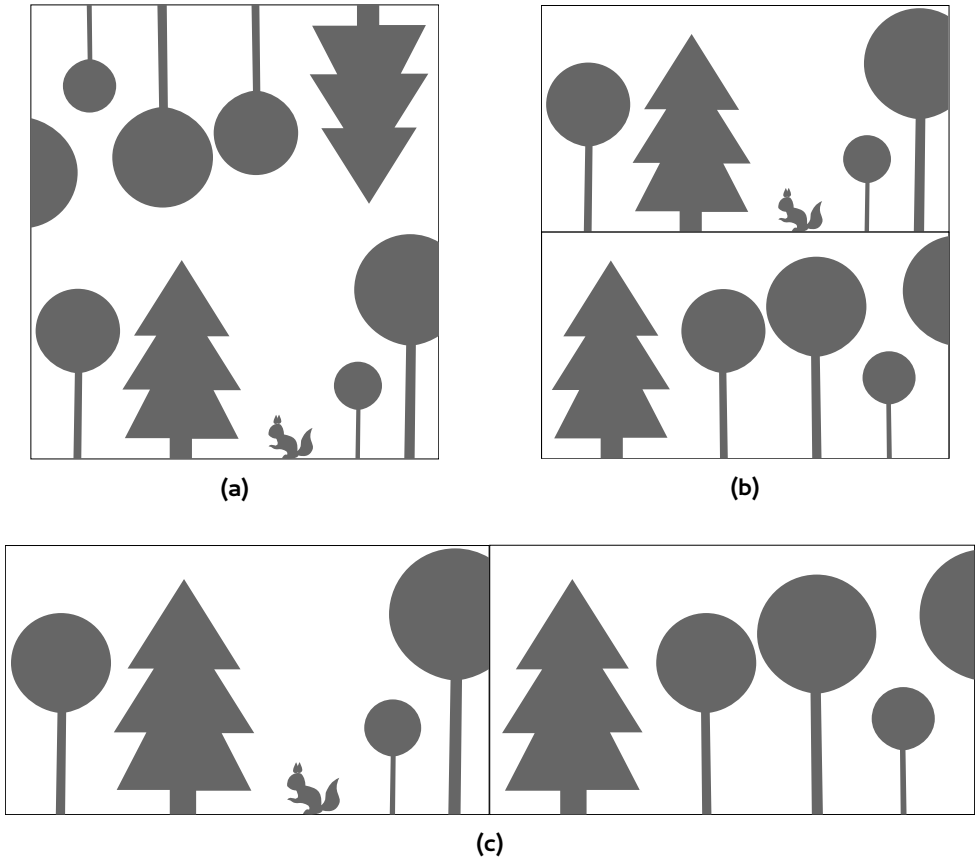


Figure 6.1: Comparison of the raster alignment as (a) expected w.r.t. the scanning process of the Z+F Imager 5006i and the Faro Focus 3D, (b) actual raster alignment in PTX export of scan from Trimble SCENE, (c) actual raster alignment in PTX export of scan from Z+F LaserControl. The actual image representations exhibit a distortion due to the mapping of the curvilinear grid to a rectilinear raster, which has been omitted in the illustration.

6.2.3 Indexed Attribute Lists

At the same time, the advantage of a raster alignment is also its drawback considering memory consumption. In a PTX file, the internal data order is maintained by insertion of zero tuples to denote invalid measurements. As a result, the number of redundant zeros in outdoor scenes is rather high. A significant amount of measurements is oriented towards the sky, which inevitably results in invalid measurements. For this reason, we propose a binary data container concept, which maintains the internal data order with little overhead in comparison to a simple binary list of the valid measurements.

Beside all valid measurements, which are often stored as unorganized point cloud data in a plain text file, the 2D raster coordinates of a data item need to be preserved. In PTX format, this information is encoded implicitly in the sequential data listing, which fills the raster in fixed row-major order. If lines that contain invalid measurements are removed, the order is irreversibly lost.

Basically, we propose to transfer the information about occupied and empty raster elements to a designated binary mask. The point list is kept in sequence as obtained from the PTX file, but all lines with invalid measurements are removed. The binary mask functions as indicator about a data item's raster position. If the raster is built up from the appended point list in fixed row-major order, only mask elements, which are true signify raster elements that are to be filled by data items.

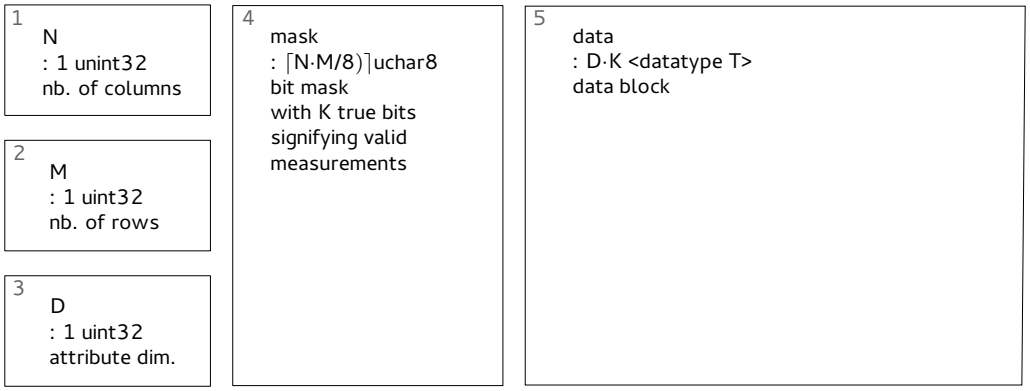


Figure 6.2: Overview of IAL format structure.

In other words, the mask is traversed in row-major order starting at the top-left corner until the current element is reached. The number of true mask elements, which are encountered before the current element, is the index of the data item in the attached list. The data item is then associated with the raster element.

The basic structure of the data container, which we call Indexed Attribute List (IAL), is presented in figure 6.2. First, two integer values in the file indicate the number of columns N and rows M of the raster. The next integer states the dimension of attributes D of a single data item. For instance, if the raster contains 3D points, each data item has $D = 3$. Consequently, the attribute dimension is not fixed to 3, but can assume other values depending on the data that is to be stored.

Since binary information occupies only 1 bit, the mask can be represented as a bit mask. $M \cdot N$ bit are necessary to store the actual mask data. K denotes the number of true bits in the mask, which concurs with the number of items in the data block. The mask bits are followed by $(8 - (M \cdot N) \bmod 8)$ bit padding to a full byte. Finally, a data block is appended, which contains the actual scan data: All data items are successively listed according to the mask in row-major order. For each data item, a space of D attributes of data type T is allocated, i.e. $K \cdot D \cdot$ sizeof(T) bytes contain the actual scan data.

In this way, the information of a PTX file can be stored efficiently, while keeping the data structure as simple as possible. The IAL data container has only little more memory consumption in comparison to a simple point list of valid measurements in binary form. In the worst case, all elements of the mask are occupied and therefore the mask information is redundant. However, the mask is usually several magnitudes smaller than the data block. Table 6.2 shows a comparison of memory consumption between the original PTX data, the point list of valid measurements as binary, and data as IAL files. In this example, the IAL files (Table 6.2 d) need at least 70% less memory than the corresponding binary PTX representation (Table 6.2 b).

Very similar in design to the presented IAL data container, is the PTG format that was developed by Leica Geosystems [Leica-2008] as well, but no resources are provided from the Leica website directly. PTG is a binary format that encodes the sequence information in a separate bit mask per scan line. Moreover, it is explicitly restricted to contain only 3D coordinates as float values.

To our knowledge, open source tools for working either with PTX or PTG are not available. For this reason, the proposed data container appears to be an advantageous contribution to TLS prototyping. In contrast to PTG, IAL has the advantage that it is easier to implement and more flexible in regard to the data that can be stored.

	Raster Size	File Type	Size on HDisk	Data Content	Raster Occupany
(a)	exported PTX file from Z+F LaserControl				
	[20238, 10000]	PTX, plain text	4.8 GB	XYZW per raster element	fully occupied
(b)	1. Clipping of raster to 7000 rows; size reduction by 30%				
	2. Limiting of range values to max. 37 m (see Section 4)				
	3. Splitting of PTX along seam into part A and B				
	[10119, 7000]	A.PTX.dat, binary	1.05 GB	XYZW per raster element	fully occupied
	[10119, 7000]	B.PTX.dat, binary	1.05 GB	XYZW per raster element	
(c)	Files containing only valid measurement data as binary				
	[16354870,1]	A.xyzw.dat	301.29 MB	XYZW per data item	list of
	[19745268,1]	B.xyzw.dat	249.55 MB	XYZW per data item	unorganized points
(d)	Separation of data layers into intensity layer and 3D coordinate layer				
	[10119, 7000]	A.ial.w	83.76 MB	W for occupied elements	occupancy: 27.9 %
	[10119, 7000]	A.ial.xyz	234.41 MB	XYZ for occupied elements	
	[10119, 7000]	B.ial.w	70.83 MB	W for occupied elements	occupancy: 23.09%
	[10119, 7000]	B.ial.xyz	195.61 MB	XYZ for occupied elements	

Table 6.2: Comparison of memory consumption of (a) original PTX file, (b) the clipped representation in binary form, (c) the total number of valid measurements as unorganized point list, and (d) the corresponding IAL representation that preserves the raster information.

IAL is intended as a tool for prototyping and does not target archiving tasks or exchange of TLS data sets. Exactly for this reason, no additional metadata is included in the format in order to provide fast I/O facilities and keep the data container lightweight and generic.

We have implemented an interface to read and write IAL files with C/C++ and Mathworks MATLAB. The condensed IAL form is for storing the file on hard disk. For processing, an IAL file is inflated into the raster representation as an array. If the raster information is not necessary, the data block can be accessed as a list of data items as done with unorganized point clouds. Both methods of data access are supported. Currently, zero is interpreted as the value that signifies empty raster elements. However, it is possible to extend the format and leave the choice of the particular value to the user.

During the development of the methods presented in the remainder of the thesis, IAL data containers have proven to be beneficial by significantly speeding up the working process. IAL is a binary file; therefore, I/O is rather fast, which is usually the bottleneck of working with plain text files. Since the data is already provided as raster if the IAL file is inflated, generation of images from raster data is a straight forward procedure. This is particularly important for evaluation as visual inspection is still the fastest way to assess results and to detect errors. For example, intensity or range can be stored as their original float values and need only to be converted to an actual image format like TIFF if necessary. Images that highlight different range intervals may be generated from a single IAL source file. Moreover, such an image gives an overview about the entire scan (A or B, respectively).

Another positive aspect of IAL is that data layers can be stored in separate files. Not every processing operation requires all layers of scan data. Often, only 3D coordinates are processed while intensity values are neglected. Therefore, it is convenient to store each data layer as a separate IAL file. As shown in table 6.2 d), intensity data and 3D coordinates can be stored as different files. Furthermore, an IAL file is not restricted to data originating in PTX files. Intermediary results from processing of raster data can be stored in this generic data container as well. For instance,

File	Content per Item	Data Type	Required Memory for IAL file	Required Memory for Deflated Array
e01.spn01.B.ial.xyz	3D coordinates	float	201 MB	811 MB
e01.spn01.B.ial.w	intensity	float	45 MB	271 MB
e01.spn01.B.ial.label	label	unsigned int	45 MB	271 MB
e01.spn01.B.ial.nn	validity of half-edges	unsigned char	18 MB	68 MB

Figure 6.3: Comparison of memory requirements for a scan of size 10119×7000 measurments from data set B. unsigned int and float are assumed as 4 byte each, unsigned char is 1 byte in size.

IAL was already used to store the hemispherical projections that were generated from TLS data by Schmidt et al. [Schmidt-2012] using a virtual camera.

As a result, separate data layers are available for each TLS scan, which are currently distinguished by file extension:

- *.ial.xyz: 3D coordinates per element
- *.ial.w: intensity per element
- *.ial.d: range per element

Table 6.3 compares the memory requirements of IAL files and their array representations if the data is inflated to the raster.

Currently, file extensions serve as a hint to the user, who has to specify the primitive data type of the IAL data block manually when reading and writing files. The information about the data type is not included in the IAL file, though the format specification could be augmented accordingly. Furthermore, the data type is fixed for the entire data block. In other words, a data tuple cannot consist of attributes that are of different data modality. It has to be stored in separated files, i.e. as separate data layers.

6.3 Method

On the basis of the raster alignment of TLS data, we can apply image processing methods. In order to gain insights about the shape of trees that are present in the considered data set, we propose to adapt CCL and boundary tracing to the specific characteristics of TLS raster data. The methods that are presented in the following lay the foundations for the retrieval of skeletal structures, which is explained in the next chapter.

6.3.1 Connected Component Labeling

In a single scan, the TLS instrument samples object surfaces that are in the line of sight of the scanner, as explained in section 2.3. Points that were measured from the same, small area of object surface are located in the same region of 3D space. At the same time, these points appear in the data raster in close proximity as well. Connected Component Labeling (CCL) can be utilized to assign a unique group label to raster elements that share a predefined property such as spatial proximity. Each group of elements is referred to as a connected component. The notion of connectedness has

to be defined specifically for the considered data set in order to decide whether a pair of adjacent elements actually belongs to the same group.

We propose to perform CCL directly on the 2D data raster that is provided as an IAL file. For this task, we employ CCL as described in [Shapiro-2001], which is summarized in algorithm 2.1. In order to process 3D TLS data with this algorithm, *connectedness* is redefined: Two adjacent raster elements are connected, i.e. most likely to be sampled from the same object surface, if the difference of attached range values is below a predefined threshold ϵ_{CCL}. If this condition holds, their attached 3D points p_i, p_j are in close proximity. The condition can be expressed as

$$abs(\|p_i - p_{scanner}\| - \|p_j - p_{scanner}\|) < \epsilon_{CCL} \tag{6.6}$$

with $p_{scanner}$ denoting the scanner position. Figure 6.4 shows the result of the modified CCL. As a result, the parameter ϵ_{CCL} controls how fine-grained or coarse the assignment of raster elements to clusters will be. If ϵ_{CCL} is very small, the resulting number of connected components will be very high. Consequently, if ϵ_{CCL} is selected rather large, the number of connected components will be small. In this case, also sets of raster elements, which were sampled from surfaces of different objects, might be joined in the same connected component.

The resulting label matrix can be stored as IAL file (*.ial.label), which is congruent to the other data layers of the same source data, and contains the label as an unsigned integer value for each raster element.

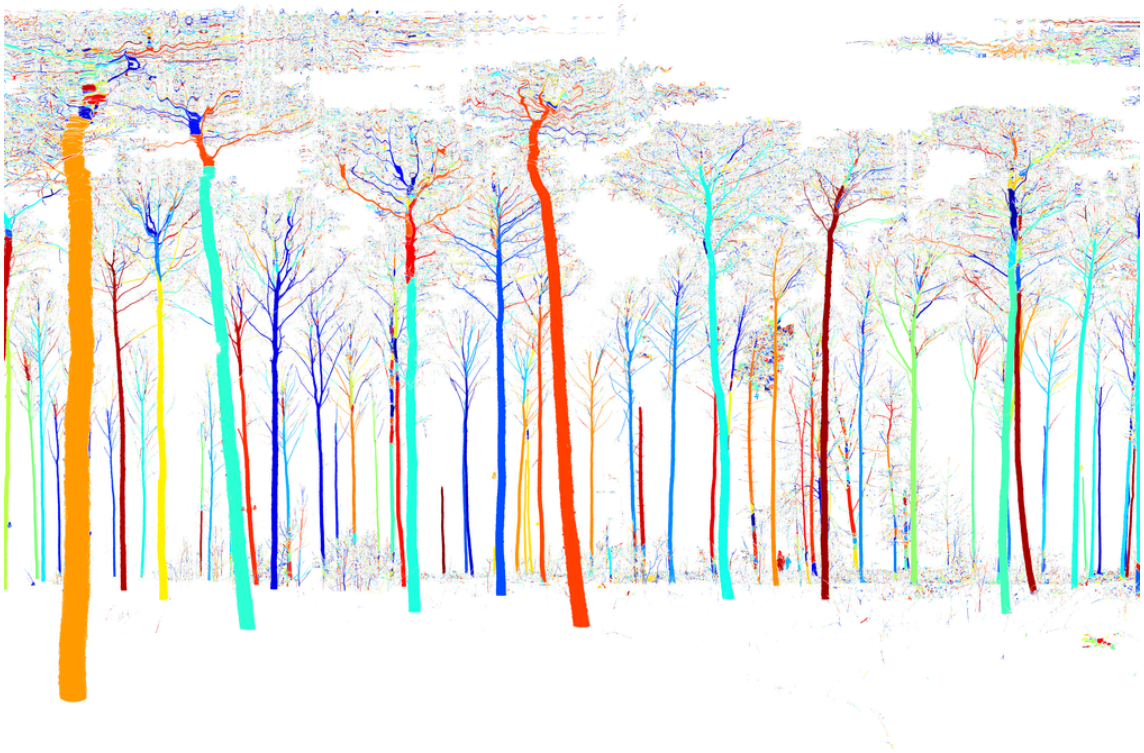


Figure 6.4: Result of applying the CCL on a scan from data set B. Labels are mapped to random colors.

6.3.2 Boundary Tracing of a Connected Component

After the retrieval of connected components, our objective is to retrieve information about a component's shape from its interior boundary. The sequence of raster elements that traces a component's interior boundary is part of the component, whereas elements of a component's exterior boundary do not belong to the component as indicated in figure 6.5. Determining the interior boundary of a component is fundamental for the retrieval of skeletal structures that is presented in the next chapter.

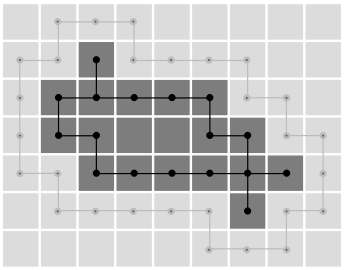


Figure 6.5: Raster elements of the interior boundary (black line) belong to the connected component (elements in dark gray), whereas elements of the exterior boundary do not.

In order to obtain interior boundary information, we construct an oriented adjacency graph from the raster alignment that is illustrated in figure 6.6a and 6.6b. Each element is a vertex. A pair of adjacent elements is connected by a pair of half-edges, which are pointing in opposite directions as depicted in figure 6.6c and 6.6d. The set of outgoing half-edges per vertex is sorted in clockwise order. The resulting configuration coincides with a rotation system of the raster if it is interpreted as a graph [Mohar-2001]. Closely related to a rotation system from graph theory is the Half-Edge Data Structure [Mäntylä-1988], which is a popular data structure for representing polygonal meshes that need to support queries about connectivity and orientation of polygons in order to apply subdivision schemes, for instance.

The label matrix, which was computed by a CCL procedure on TLS range data, is the basis for the construction of the rotation system. Edges between vertices of different labels are marked as invalid edges. Since, each element in a 2D raster with 8-adjacency has 8 outgoing half-edges, the information about the individual validity of all 8 half-edges can be stored as 1 byte per raster element. As a result, a new IAL file (*.ial.nn) can be generated, which represents the rotation system. We enumerate half-edges in clockwise order starting with the North pointing half-edge in the least significant bit, as demonstrated in figure 6.7.

If traversal is started along a half-edge of an arbitrarily selected inner vertex and continued by advancing always along the next outgoing half-edge in clockwise order, the result is a round-trip path, which is called an atomic cycle by Klette and Rosenfeld [Klette-2004], as highlighted in figure 6.6e. In a rotation system of a 4-adjacency raster grid, an atomic cycle always traverses 4 edges in counter-clockwise order. Analogously, in a rotation system of an 8-adjacency raster grid, the corresponding atomic cycle traverses 8 half-edges and results in a self-crossing path as demonstrated in figure 6.6f. Only the boundary cycle is traversed clockwise. In 4-adjacency, the boundary of inner holes is traversed counter-clockwise but longer than an atomic cycle. In 8-adjacency, the boundary of inner holes can be recognized because the path is not self-crossing and also counter-clockwise.

Cycles that are relevant for describing shape features of the component in the label matrix can be easily recognized by their traversal order and path length. The sequence of interior boundary elements of a connected component with 8-adjacency can be efficiently retrieved by traversing the boundary cycle as summarized in algorithm 6.2. First, the raster is traversed in row-major order

starting at the top left corner until an element with the label in question is encountered. Therefore, the direction from which this element, i.e. the vertex was entered, is known. The opposite half-edge corresponding to the entering direction is determined and the next valid half-edge in order is determined and traversed. The already traversed half-edge is marked as visited. Basically, this procedure is repeated until the first element is visited another time and the next clockwise half-edge is found to be already visited. The algorithm is also described by Gross and Tucker [Gross-1987]. However, we suppress self-crossings by marking half-edges as visited if their traversal would result in a self-crossing (algorithm 6.2, lines 9-20).

The result of the boundary tracing is a sequence of raster elements that traces the component silhouette. During boundary tracing, a raster element may be visited several times, but entered

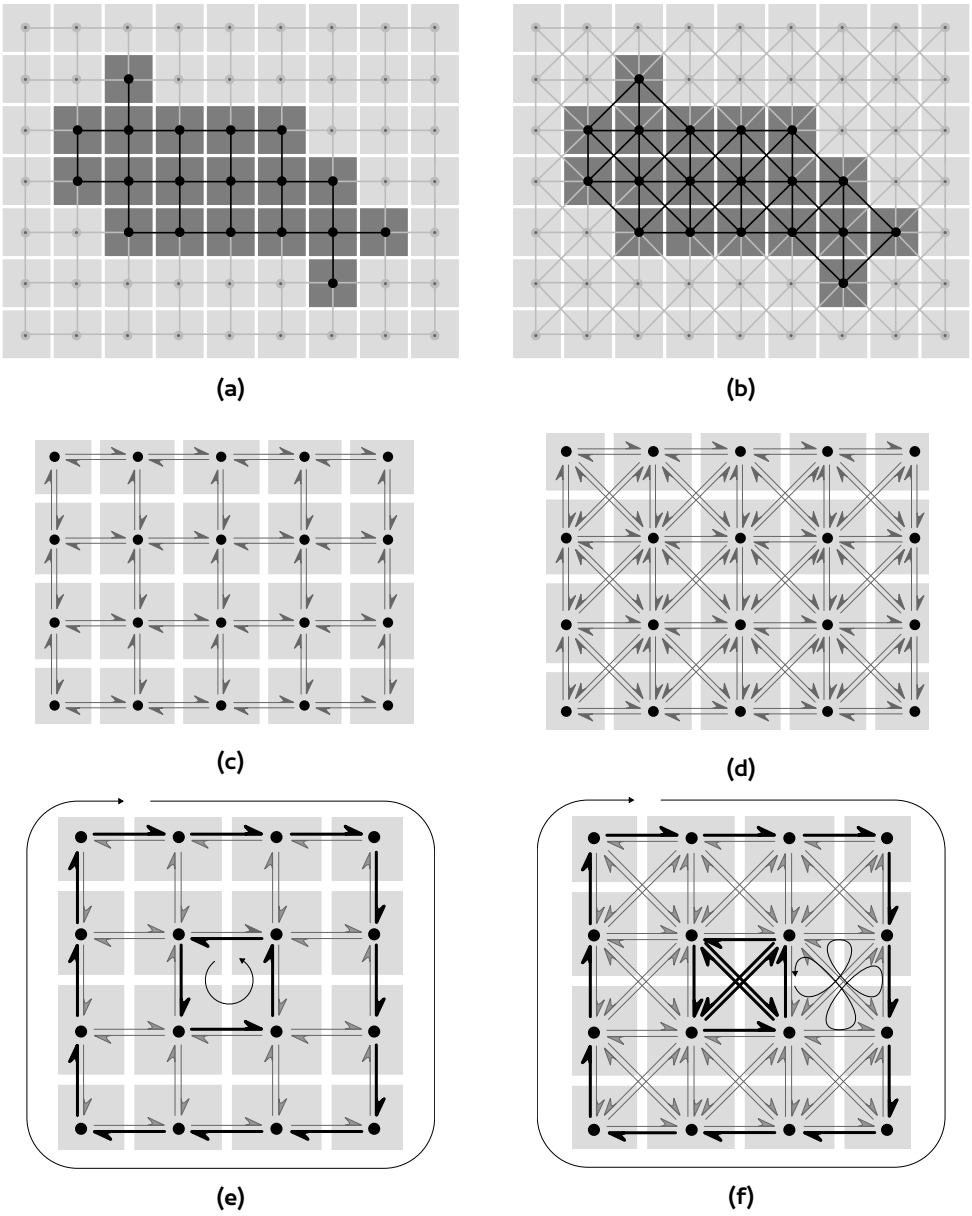


Figure 6.6: The raster is interpreted as graph with (a) 4-adjacency, (b) 8-adjacency. Resulting rotation system with two opposite pointing half-edges per edge in (c) 4-adjacency and (d) 8-adjacency. Atomic cycle in (e) 4-adjacency, and (f) 8-adjacency. Boundary cycle is always traversed in clockwise order.

Algorithm 6.2 Cycle tracing using a rotation system of the raster with 8-adjacency

```
1  procedure TRAVERSE((x, y): 2D raster coordinates, h_dir: entering direction to access (x, y))
2      P ← ∅                                   ▷ Initialize sequence of boundary elements as empty.
3      prev_1 ← ∅
4      prev_2 ← ∅
5      while isUnvisited(x, y, h_dir) do
6          isUnvisited(x, y, h_dir) ← false
7          P ← P ∪ (x, y)
8          (x, y) ← move along half-edge h_dir to next element
9          if prev_1 = N and prev_2 = SE then
10             hasHalfEdge(x, y, SW) ← false
11         end if
12         if prev_1 = E and prev_2 = SW then
13             hasHalfEdge(x, y, NW) ← false
14         end if
15         if prev_1 = S and prev_2 = NW then
16             hasHalfEdge(x, y, NE) ← false
17         end if
18         if prev_1 = W and prev_2 = NE then
19             hasHalfEdge(x, y, SE) ← false
20         end if
21         h_opp ← opposite half-edge to h_dir
22         for i ← 1 . . . 7 do
23             h_next ← (h_opp + i) mod 8
24             if hasHalfEdge(x, y, h_next) then
25                 h_dir ← h_next
26                 break
27             end if
28         end for
29     end while
30     return P
31 end procedure
```

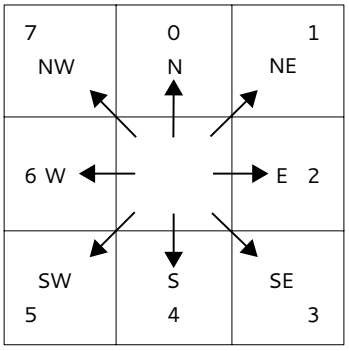


Figure 6.7: Adjacency of a raster element is named with compass directions. Validity of the half-edge towards each of the 8 adjacent elements is encoded in a byte. Each direction corresponds to a bit, starting with the North element in the least significant bit.

from different directions. As a result, a partial sequence of boundary elements may form a closed loop, i.e. a polygon, as illustrated in figure 6.8b. If the polygon does not directly share a raster element with another polygon, both are connected via a polyline bridge, where the boundary tracing visited exactly the same raster elements in both directions like in figure 6.8c. In addition, boundary perturbations, i.e. appendices, which do not bridge between two boundary loops, may occur as shown in figure 6.8d.

Boundary tracing on a connected component in the label matrix retrieves only the component's silhouette boundary. The strategy is closely related to the method described in [Sonka-1998]. Information about inner rifts, which are caused by depth discontinuities in a component's interior, is not included in the label matrix, as illustrated in figure 6.9.

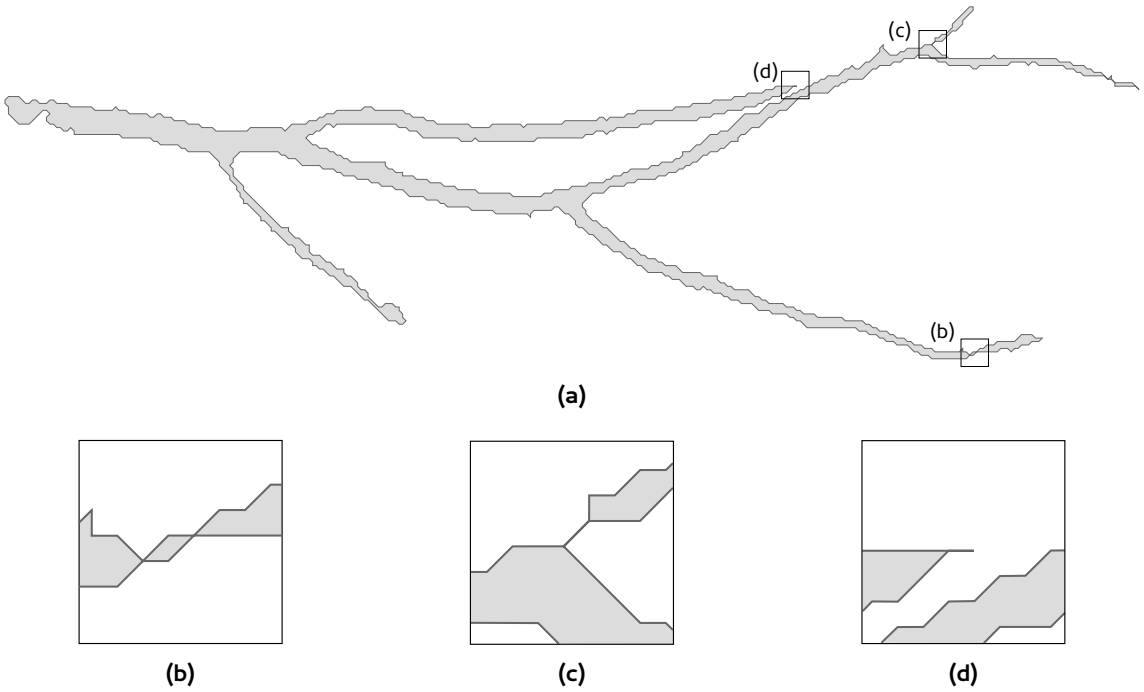


Figure 6.8: The resulting component boundary in (a) consists of three types of shapes: Boundary elements can form (b) a closed polygon, (c) a bridge between polygons, (d) an appendix.

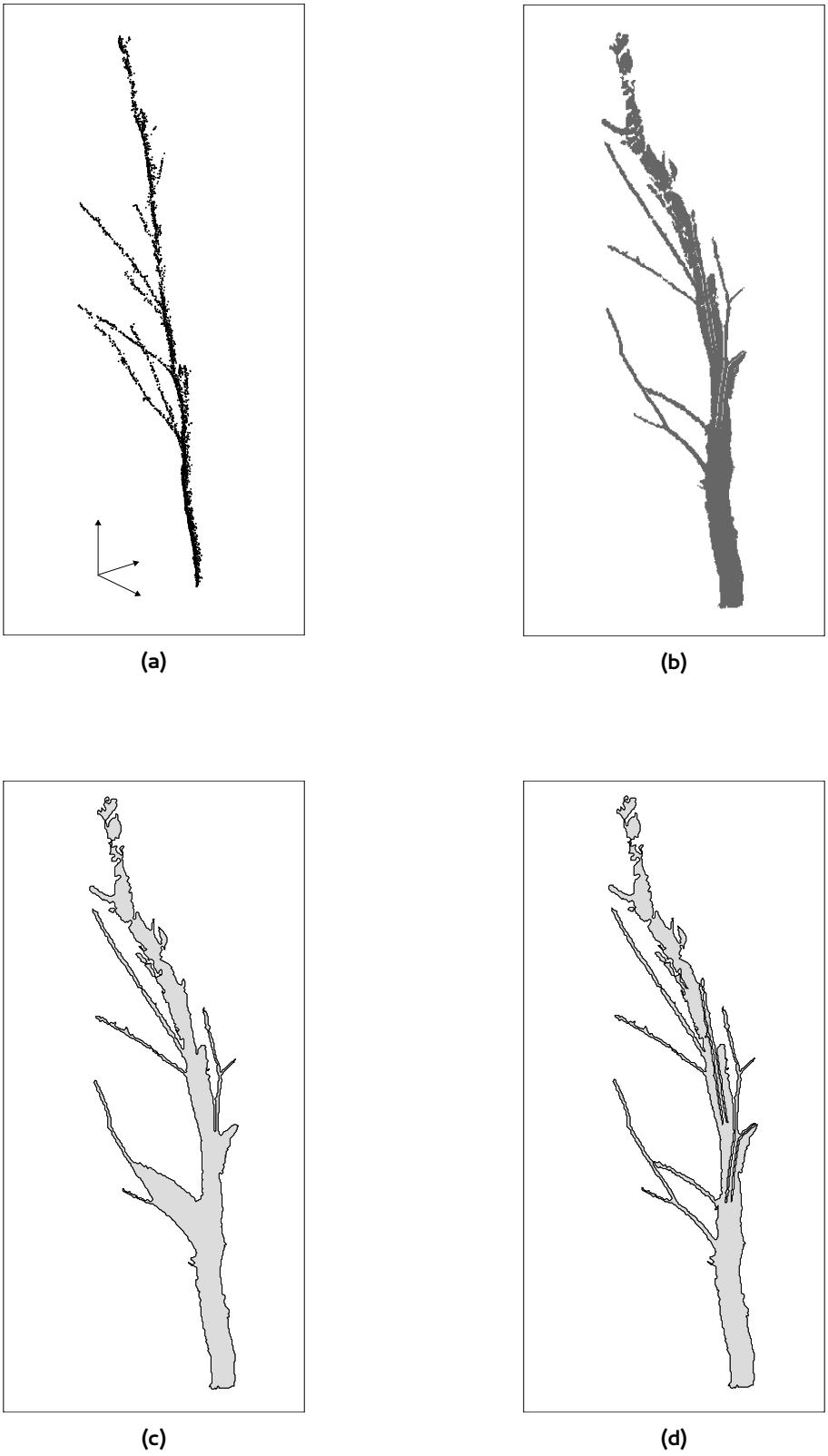


Figure 6.9: Comparison of the connected component and resulting boundaries: (a) 3D point set that corresponds to (b) the connected component in the raster. The traced component silhouette in (c) lacks contours of inner holes and depth discontinuties of branches, in contrast to (d) the tracing of all contours including inner rifts.

6.3.3 Boundary Tracing Including Inner Rifts

In a connected component, a pair of raster elements is mutually reachable via a path of raster elements. For each pair of adjacent elements equation 6.6 holds. However, not every single pair of adjacent raster elements that is part of the component fulfills condition 6.6. Range values of two adjacent elements may be very different, even though both elements are part of the same connected component. In this case, there is a rift between both elements as they are apparently not nearest neighbors in 3D space. In other words, information about the connectedness of a particular element regarding its 8-adjacency is relevant for recognizing depth discontinuities. A similar suggestion was formulated by Cheng et al. [Cheng-2007].

For this reason, we propose to evaluate the validity of half-edges in the rotation system on the basis of the TLS range data. The IAL file (*.ial.nn) is generated as explained previously, but for each half-edge condition 6.6 is evaluated separately. As a result, a half-edge is only valid if the two vertices have the same label and if their corresponding range values fulfill condition 6.6. In this way, inner rifts within a component that are caused by depth discontinuities in the range data are reflected by the rotation system.

Boundary tracing is performed as in algorithm 6.2 but on the updated rotation system. Consequently, inner rifts that share vertices with the interior boundary are traversed as well as shown in figure 6.9d. Therefore, the resulting sequence of boundary elements reflects the shape of the point cloud in 3D as seen from the scanner position more precisely than the component's silhouette boundary.

6.3.4 Tracing of Hole Boundaries and Inner Rifts

Frequently, a component has holes, i.e. groups of elements which are entirely surrounded by component elements but do not belong to the component themselves. The group of elements may be a separate connected component. But more likely values are simply missing and the group appears as a data gap. Tracing of holes is essential to determine a component shape precisely. For example, if holes are not determined in the crown component, the component shape might be modeled insufficiently.

As indicated in figure 6.9b, a branch may point towards the scanner position. In the 2D view of the raster alignment, there may be only comparably small holes at the location or no holes at all. But the inner rift that is caused by such a depth discontinuity is included in the rotation system. Similarly, modeling of such inner rifts, which are embedded in a component, is vital.

As a consequence, we trigger a traversal at each vertex, which has a degree less than 8 in 8-adjacency. In other words, if the vertex is not an inner vertex and consequently not connected to all of its neighbors, it is located near a hole or an inner rift. Naturally, traversed half-edges are marked as visited such that a particular path can be traversed only once.

6.4 Experiment Setup and Results

We have conducted experiments with all three data sets B, K, and E. The representation of a tree in data set B consists of data from at least two viewpoints. In other words, a tree data set actually comprises several IAL files. Each IAL file contains scan data within a cylindrical bounding volume at the given tree position from the respective viewpoint, as shown in figure 6.10. Some of the regions of interest may be overlapping because neighboring trees were selected. In data set E,

trees were not detected in advance and the data of the entire viewpoints that is depicted in figure 4.5 was processed directly.

In order to generate IAL files from PTX source files, tools were implemented to perform the required conversions. As stated in section 4, the range was limited to 37 m for each viewpoint. Moreover, it has to be noted that measurements in the PTX file may have 3D coordinates even though the intensity value equals zero. Though, we exclusively consider measurements with intensity greater than zero for further processing.

The proposed algorithms were implemented in C++. Eigen [Eigen] and Boost [Boost] libraries are employed as well. All experiments were computed on a dual-core machine (4 GB RAM, Linux, 64 bit). IAL files were utilized for input and output operations of data sets.

The result of the CCL procedure on a tree of data set B is demonstrated in figure 6.11. For each raster element in the source file containing 3D coordinates, a label is assigned, which indicates its association with a particular connected component. The raster is stored as *.ial.label file on disk. At the same time, a *.ial.nn file is created, which stores the corresponding rotation system that has been created on the basis of the range values.

CCL was performed on all data sets with $\epsilon_{CCL} = 0.05$ m. The threshold was set after empirical testing. It is crucial that range values, i.e. the Euclidean distance of a 3D point to the scanner is used for condition 6.6 in the CCL procedure. If only the Euclidean distance of a point's projection onto the XY-plane to the projection of the scanner position is considered, elements that are definitively not located on the same object surface are joined in one component. Considering the scan situation of the Z+F Imager 5006i, a large amount of scan points is contained in a conical space above the scanner's position. If distances on the XY-plane are used for CCL, basically all those points will be joined in one connected component. However, those points exhibit strong variation in Z-coordinates. Therefore, only distances in 3D space should be considered for CCL computa-

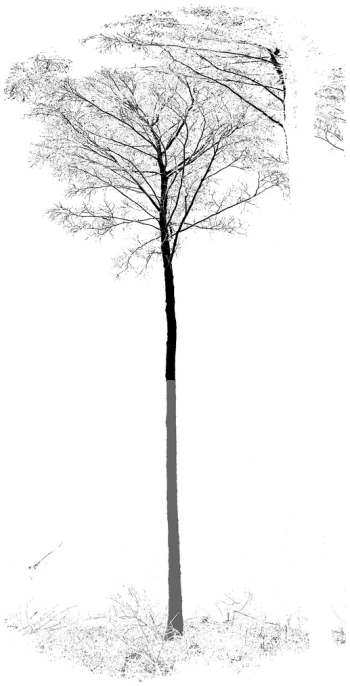


Figure 6.10: Mask of an IAL file representing a tree data set. The 3D points have been restricted to a cylinder around a given tree position. Only black regions are considered in the experiments. Gray regions have been omitted.

Figure 6.11: Resulting label matrix after applying CCL to a Birch tree from data set B. Labels are mapped to random colors.

tion. The generation of the *.ial.label and *.ial.nn files on the basis of the input *.ial.xyz file is rather fast and takes about 0.01 to 5.04 seconds.

The tracing of boundaries, inner rifts, and holes was performed on the label matrix (*.ial.label) and the rotation system (*.ial.nn), which were generated in the previous step. For the experiments, the processing was limited to connected components that are between 1,000 and 100,000 pixels in size. The algorithm does not rely on any parameters that have to be set. Furthermore, no significant additional memory is allocated beside the memory space needed for the input data sets and an additional array of same characteristics as the rotation system in order to mark half-edges that were visited. As already pointed out in table 6.2, the memory consumption of *.ial.label, and *.ial.nn files is depending on the raster size and occupancy, but in comparison to the original 3D data set rather low. The processing is rapid: Boundaries with inner rifts and holes of 2543 components are calculated in 7.8 min in total. Naturally, the processing time depends on the component size and the number of atomic cycles that have to be tested. Some processing results are demonstrated in figure 6.12 and 6.13. The output of the algorithm is a sequence of 2D raster coordinates. Each item has a label that indicates to which sequence – either main boundary or hole – it contributes.

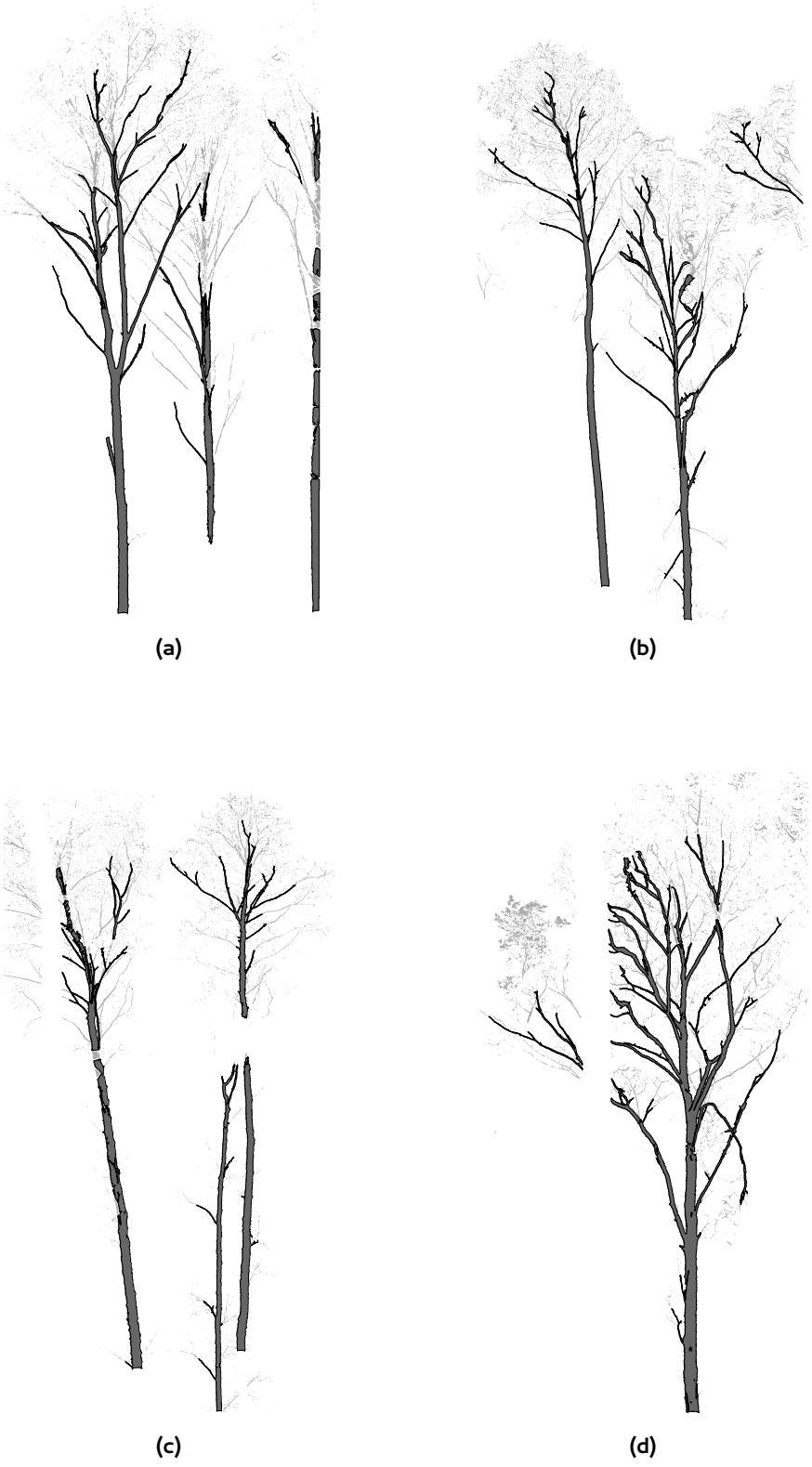


Figure 6.12: Results of the experiments on data set B. Components that have been processed are drawn in dark gray. Retrieved boundaries are drawn as black lines. Components in light gray have been omitted in the experiments. The tree in subfigure (d) is the same used in figure 5.5a in chapter 5.

Figure 6.13: Results of the experiments on data set P in subfigure (a) and (b), and on data set E (c). Components that have been processed are drawn in dark gray. Retrieved boundaries are drawn as black lines. Components in light gray have been omitted in the experiments. A case when the main boundary has not been found is visible in the right section of subfigure (c).

6.5 Discussion

As pointed out before, the dependence on a particular, moreover proprietary format as PTX is arguable. Though, it seems to be a common format to exchange TLS datasets between software. Therefore, it can be assumed that it will not become deprecated anytime soon.

In any case, it is worthwhile to investigate the raster alignment that can be recovered either from a PTX file or from the measured spherical coordinates as the experiments have shown. Generally speaking, a scan in raster alignment seems to be a more natural way to organize the 3D data in contrast to a voxel space that imposes an arbitrarily defined structure onto the scan data. However, this organization is then restricted to a single scan.

The presented data management concept in form of IAL was created as a tool to provide fast I/O facilities and a simple but efficient way to store the TLS data as well as interim results. Plain text files, which are often used, are slow considering I/O operations: Character sequences need to be parsed and converted to their corresponding numeric binary representations. Although this is a native feature of the programming language, these operations are very time consuming for large data sets. However, simple binary dumps of the same data give rise to errors, e.g. when data is interpreted not as intended because the lack of additional description or meta data.

There are efforts to establish international standards for laserscanner data formats, e.g. LAS [LAS-2011] or E57 [Huber-2011]. But they are chiefly targeting ALS data. In addition, these formats include comprehensive meta data in their specifications for each data set, which is beneficial for achieving and exchange. Though, the same feature makes it rather unsuited for development and prototyping tasks. Plenty of processing operations do not require the provided meta data at all. For this reason, a lightweight, generic data container is of advantage. IAL files turned out to be very handy during the research work presented in the thesis. Several extensions to make the IAL files and working with them more user-friendly are possible.

CCL on range images is not a new invention, but was performed before. For instance, Bienert and Schneider [Bienert-2013] mapped a limited interval of range values to gray levels and afterwards performed CCL. The major difference to the method as presented in the thesis is that TLS data is evaluated directly. Since the data is not mapped to an intermediary gray level image, there is also no need to enforce restrictions considering the range values. Visualization of the input or output data sets is an operation, which is separate from the CCL process.

As a result, the entire content of a scan can be processed at once. Moreover, only a single parameter is presently needed for the CCL procedure. The threshold ϵ_{CCL} decides whether two adjacent raster elements are connected, i.e. whether their range values are similar enough to be sampled on the same surface. Since the parameter is fixed for a data set, there are regions that are under- or oversegmented in the resulting partition. Often, large parts of ground surface are joined with raster elements representing trunk surface in the same connected component due to the spatial proximity. In the experiments with data set B, a high amount of very small connected components can be found in the tree crowns. It has to be emphasized that data set B was acquired in winter, when the trees were in leaf-less state. Consequently, it is unclear why there are so many, apparently noise points present in the tree crowns. We consider this to be a specific issue of the employed TLS instrument. Clearly, it would be advantageous to replace the single parameter ϵ_{CCL} with a mechanism that takes the beam divergence and consequently the increasing inter-point distances in dependence of the range values into account. Recently, Bienert and Schneider [Bienert-2013] already proposed such an approach that could be adapted to our method as well.

A rotation system [Gross-1987] on the basis of the 8-adjacency graph of a label matrix from CCL can be studied regarding its topological characteristics and the result of traversing the graph as described is clear. However, if the rotation system was manipulated by evaluating equation 6.6

for each half-edge as proposed, the outcome of traversal is not necessarily clear in advance. The validity of both half-edges that compose one full edge is determined simultaneously. In other words, a half-edge has always the same state as the corresponding opposite half-edge. But since the range values are depending on the particular object, scanner and possibly noise effects, the resulting rotation system might include odd configurations of edges. The traversal of successive half-edges may eventually block itself. In the conducted experiments and excessive testing during the development of the method, this occurred only rarely and appears to be connected to strong distortion effects due to wind. In this case, normal boundary tracing on the label matrix can be considered as a fallback solution. Clearly, this issues needs to be investigated further.

As demonstrated in figure 6.9d, component boundaries that are enhanced by descriptions of inner rifts represent the discernible shape of the point cluster in 3D space more closely than the silhouette contour, which may lack significant features. The locations of half-edges that are part of an interesting cycle instead of an atomic cycle in the rotation system are unknown. They have to be detected by a linear search over all raster elements. Similarly, the relevance of a particular cycle can only be assessed after completing it. However, these considerations can be neglected in practice because processing is very fast. Besides the memory that is occupied by the input data (*.ial.nn, *.ial.label), only a copy of the rotation system is needed to mark visited half-edges during the procedure. An explicit graph structure for the rotation system is not constructed. Instead the algorithm operates directly on the array data, which makes it particularly efficient. Moreover, no parameters have to be set to control the algorithm performance.

In the experiments, the retrieved boundary descriptions in general concur with the boundaries that can be recognized by visual inspection of the rotation system (*.ial.nn). It is important to realize that the inner rifts that are caused by branches, which grow out of the trunk towards the scanner position, are not recognizable if the connected component in the label matrix is inspected. In fact, this information is not present in the label matrix at all. For this reason, inner rifts cannot be retrieved by a standard approach to boundary tracing on raster images. The information about inner rifts that originates in the range data is exclusively embedded in the rotation system.

In 2D space, the resulting boundary contour is rather ragged. Especially trunk boundaries have plenty of perturbations. In 3D space, the corresponding polyline is equally jagged. A connected component naturally has a main boundary and possibly a number of hole boundaries. Presently, we have no definitive mechanism to decide whether a particular hole boundary is solely the result of missing data or a topologically relevant feature. The area of a hole or its boundary length could be used to decide this question. But similar to previously discussed parameter values, it does not seem adequate to set an absolute threshold because the component sizes vary strongly within a scan.

In the experiments, we found that there are differences between the scan data of the two TLS instruments, which has a major impact on the quality of the results. As shown in figure 6.14, the Faro Focus 3D, which was used to scan data set E, samples 3D points between a branch and the trunk surface. On close visual examination, these points seem to be floating in the air. Most likely, they are a result of averaging the range values of an emitted laser beam that caused several reflections. In any event, the sequence of spurious points is reflected in the rotation system as well and prevents the tracing of the rift between branch and trunk surface to the full extent. Therefore, the boundary description cannot capture all of the relevant features, which are discernible in the point cloud. We have not noticed similar issues with data that was acquired with the Z+F Imager 5006i. Clearly, this example demonstrates that the differences between TLS instruments have to be considered and eventually compensated. Moreover, it proves that methods that were proposed in the literature have to be assessed w.r.t. the used input data and TLS device, which must not be neglected.

Figure 6.14: The Faro Focus 3D samples points between branch and trunk surface that are apparently floating in the air and do not represent actual surface samples. These spurious points hinder tracing of the rift between branch and trunk to the full extend.

6.6 Future Work

Besides the improvements that were already pointed out, the data from the PTX file could be used to estimate normal directions as it was its original purpose. On the basis of the label matrix, point neighborhoods for fitting surface patches in order to compute normal directions can be readily determined. Furthermore, curvature parameters could be retrieved as well. The data could help to differentiate hole boundaries in descriptions of shape features and contours of data gaps, which could be eventually discarded.

Up to now scans have been processed separately. But one of the most important aspects of laser-scanning objects is the full 3D representation of objects. For this reason, components and their boundaries from different viewpoints, which describe different parts of the same objects, need to be somehow combined accordingly. After extraction of relevant components and boundaries, the voxel space may provide an appropriate framework to recognize neighborhood relations between established connected components. The combination of element-wise analysis in the raster alignment and the evaluation of spatial relations on a larger scale in the voxel space appear to be a promising approach.

7 RETRIEVAL OF SKELETAL STRUCTURES FROM A TLS SCAN

In this chapter, we present a novel approach to retrieve skeletal structures from single scans. The key idea is that an intermediary 2D skeleton is generated and utilized to reconstruct the 3D skeleton of a connected component. Connected Component Labeling (CCL) and boundary tracing as explained in the previous chapter are essential for the presented method. Finally, we present experiment results and discuss the method in detail.

7.1 Motivation

The aim of skeleton retrieval is to generate an accurate and compact representation of an object's geometry. However, the method presented in chapter 5 provides only an approximation of the object's expected skeleton. The level of detail that can be reconstructed strongly depends on the resolution of the voxel space and the particularities of the input data set.

With the insights gained regarding the scanning process and neighborhood relations, we have come to the conclusion that the task of skeleton retrieval should be approached from a different side. As a result, we have developed a novel strategy for retrieving skeletal structures from connected components that represent parts of a tree. The method produces partial skeletons that are of higher detail in comparison to the skeletons, which can be retrieved from voxel space. Our strategy relies on the raster alignment of TLS data (see section 6.2), CCL (see section 6.3.1), and boundary tracing including inner rifts (see section 6.3.3). Thereby, it represents an immediate continuation of the work presented in the previous chapter.

Generally speaking, the automatic determination of a tree skeleton from a 3D point cloud that had been segmented manually is basically doable. A cluster of 3D points may be distributed among an invisible, yet discernible space curve. Hence, it most likely represents a tree branch. But how to obtain at least an approximation of this invisible space curve is a challenging task. Hastie [Hastie-1984] and Hastie and Stuetzle [Hastie-1989] were the first to introduce the notion of summarizing such a point distribution as a space curve, which they called a Principal Curve. However, their definition of Principal Curves makes computation in practice too cumbersome according to [Kégl-1999]. Kégl [Kégl-1999] revised the definition of a Principal Curve and introduced the Polygonal Line Algorithm (PLA) that computes a descriptive polyline for an input set of 3D points.

We have conducted initial experiments in order to assess, whether the PLA is applicable to TLS data [Schilling-2012a]. For this first evaluation, individual trees were isolated in single scans of data set B. Trees were manually segmented into point subsets that represent branches or trunks. For each determined subset of 3D points from the single scan, a Principal Curve was computed

by the PLA. We found that the PLA [Kégl-1999] is a suitable tool to obtain a summarizing polyline for a set of 3D points from TLS data.

For this reason, the focus of our research was to replace the manual segmentation of a single scan with an automatic procedure. Similarly to the preliminary experiments, our starting point for performing segmentation of a single scan is the label matrix. A label matrix, which is determined by CCL, provides an initial partitioning of the data set. The applied CCL procedure is explained in section 6.3.1. Henceforth, our approach to skeleton retrieval is founded on the assumption that the topology of a connected component in a label matrix concurs with the discernible topology of the corresponding 3D point set. If the connected component is interpreted as a graph, the difference between its 2D and 3D geometry is basically the displacement due to range values since the grid-like alignment is preserved.

7.2 Method

The key idea of the approach is that the skeleton of a component in the 2D raster arrangement can be used to segment the 3D point set into clusters, which represent individual branches or trunk parts. Then, point subsets can be subjected to the PLA to obtain a compact description of the component as a set of 3D polylines, which is possible because the relation of 2D raster coordinates to 3D coordinates is given.

In the following, the term skeleton concurs with skeletal structure and means the description of the main shape features of the considered connected component in the label matrix as a line graph either in 2D or 3D. Consequently, we aim at retrieving skeletons from single TLS scans only. Combination of several skeletal structures to represent a complete tree skeleton is considered future work.

7.2.1 2D Skeleton Approximation via Voronoi Diagram

As explained in section 6.3.3, the boundary information of a connected component can be efficiently retrieved from the label matrix in combination with a rotation system that takes the measured range values into account. The 2D skeleton of the considered connected component can then be determined straightaway from the Voronoi diagram [Aurenhammer-2000].

We calculate a Voronoi diagram from the set of boundary elements B of a connected component C. B comprises the elements of the component's silhouette contour, the elements of traced inner rifts, as well as boundaries of embedded holes and their attached rifts. Figure 7.1 shows the resulting Voronoi diagram.

Only Voronoi edges, which are entirely contained within the component's boundary, contribute to the Medial Axis approximation of the component. The set of relevant Voronoi edges can be identified by testing each one whether its Voronoi vertices are located within the boundary polygon. The resulting initial skeleton estimate is illustrated in figure 7.2. If hole boundaries are included in the computation of the Voronoi diagram, the skeleton graph is not necessarily a tree graph. Most often, holes are reflected directly and cause cycles in the skeleton graph.

As shown in figure 7.2, the initial skeleton estimate might consist of several disjoint sub graphs. Naturally, an adequate skeleton of a connected component can only be represented by a connected graph, where each two graph vertices are reachable. Consequently, missing edges have to be added to the skeleton estimate.

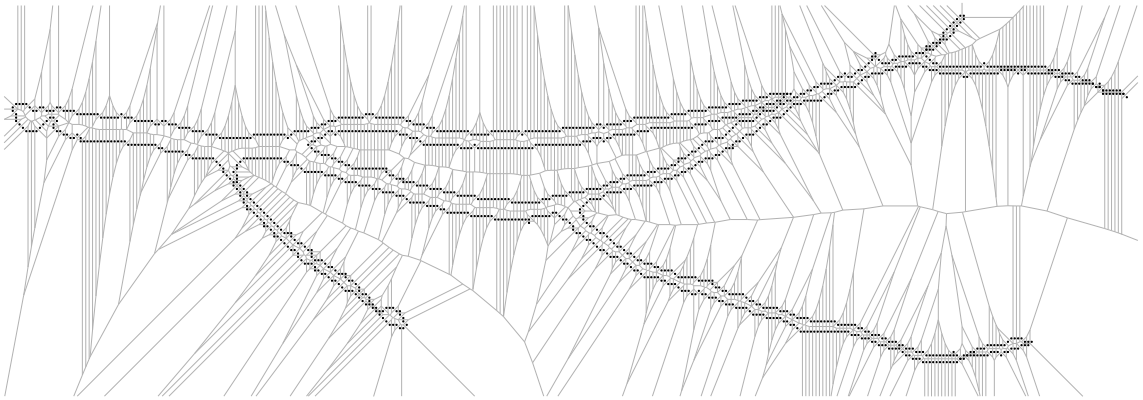


Figure 7.1: Clipped Voronoi Diagram from the set of boundary points from a connected component that is part of a scan from data set B. The original component is rotated by 90°.

Refinement of the initial 2D skeleton

Separation of sub graphs occurs if the boundary forms a corridor that is too narrow as for instance in figure 7.2a. In addition, if the boundary forms a closed loop, i.e. a polygon, which is only connected by a bridge to another polygon, both contained skeleton parts are separated as in figure 7.2b. If the area of a boundary polygon is too small as in figure 7.2c, it will not contain a Voronoi edge. For this reason, completion and refinement of the obtained 2D skeleton is mandatory prior to segmentation.

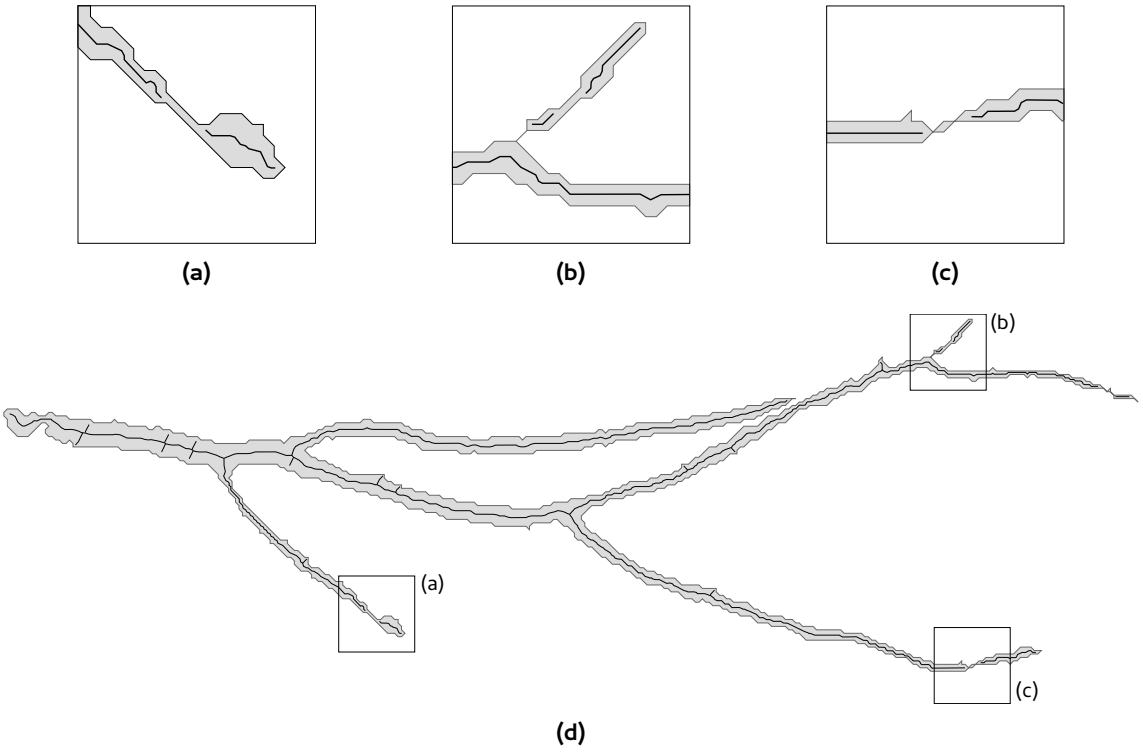


Figure 7.2: The set of Voronoi edges constitutes a skeleton that might consist of several disjoint sub graphs. Disconnection of sub graphs occurs due to (a) narrow corridors, (b) bridges in the boundary, or (c) small boundary polygons.

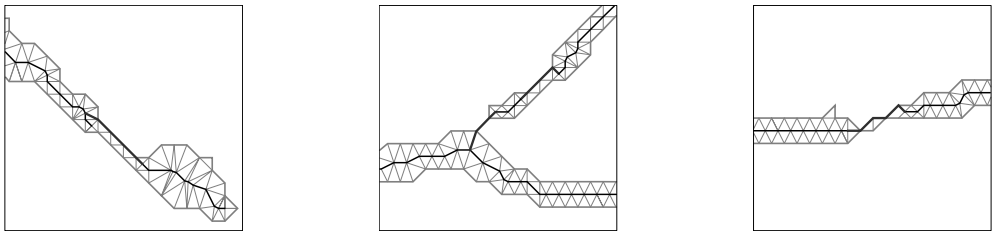


Figure 7.3: Linking of sub graphs is done on a Delaunay triangulation as search space using breadth-first search [Russell-2010]. Compare with figure 7.2d.

Addition of a straight line segment in order to connect the nearest vertices of two disjoint sub graphs would be a trivial solution. In case of figure 7.2c, the connection edge would intersect boundary edges. Consequently, parts of the skeleton graph would be located outside the boundary, which is inappropriate for the targeted segmentation. Therefore, we propose to link all sub graphs together to form a single skeleton graph by iteratively searching suitable connection paths.

As basis for searching, we compute a Delaunay triangulation, which is constrained and confirming to the Delaunay condition [Shewchuk-1996], as an interim support structure. The Delaunay triangulation is constrained because all boundary edges and skeleton edges are retained. In order for the triangulation to maintain the Delaunay condition for each triangle, input edges may be subdivided and new vertices are inserted. Holes in the boundary are also holes in the Delaunay triangulation.

As long as there are disjoint sub graphs, we pick one of them. For each of its vertices, we search a path along the Delaunay triangulation to another sub graph as depicted in figure 7.3. All vertices that belong to other sub graphs are interpreted as goal nodes. Traversal of the graph given by the Delaunay triangulation is performed in breadth-first order, which consequently finds the shallowest goal node [Russell-2010]. If a vertex of another sub graph is encountered, the traversal is terminated. The shortest path of the found ones that connects the considered sub graph to another sub graph is added to the skeleton. If paths are of equal length, the path that connects to a skeleton vertex of degree one is preferred. After this procedure, the resulting skeleton consists of a single connected graph that may share some edges with the boundary.

Connected components determined from CCL on TLS 3D data frequently have rather noisy, ragged boundaries. Especially, components representing trunk surface patches often have plenty of perturbations in their boundary as pointed out in section 6.5. Without additional information, it is infeasible to decide whether a boundary perturbation is only noise or a twig that has been sampled by only few points. As a result, a significant number of spurious edges are part of the initial skeleton estimate as indicated in figure 7.4. Clearly, filtering of the skeleton estimate is a necessary step.

Among three other variants, Ogniewicz and Kübler [Ogniewicz-1995] proposed a Potential Residual function that can be used to filter edges of a Voronoi diagram. Basically, a Voronoi edge e is weighted by the distance along the boundary B between the two boundary elements b_U, b_V $(U < V)$, which generated e. The Potential Residual can be calculated by

$$\text{dist}^B(b_U, b_V) = \min(W_U^V, W_0^{|B|-1} - W_U^V) \tag{7.1}$$

with

$$W_U^V = \sum_{i=U}^{V-1} \|b_i - b_{i+1}\| \tag{7.2}$$

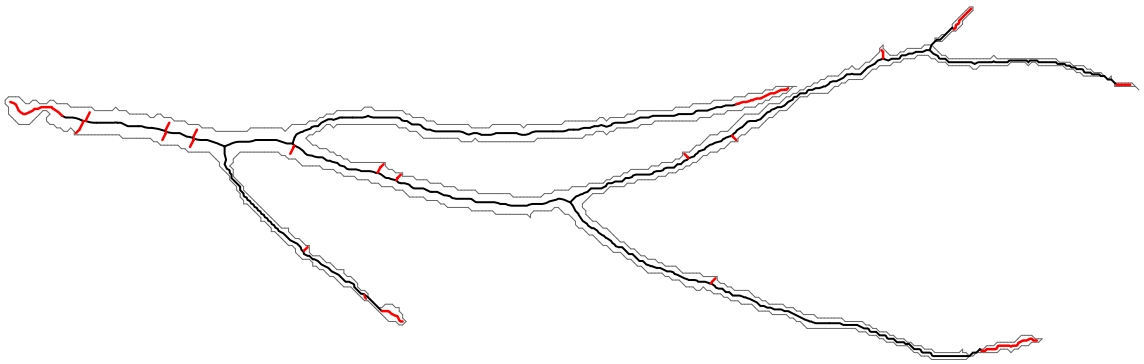


Figure 7.4: The skeleton estimate has a number of spurious edges marked in red, which are pruned using the Potential Residual [Ogniewicz-1995].

Voronoi edges that have a weight below a predefined threshold are assumed to be spurious edges, which do not contribute to the component's skeleton. In order to ensure the connectedness of the skeleton graph, we employ an iterative elimination scheme of Voronoi edges. In each iteration, only Voronoi edges that are incident to skeleton graph leaves are checked whether their weight is below a threshold ϵ_W. The procedure is summarized in algorithm 7.1. In order to ensure that the skeleton graph is not entirely eliminated in case the component is rather small, filtering is stopped if the entire skeleton graph has only two vertices that are leaves. Consequently, further refinement of such a component is skipped and the processing proceeds with the computation of a Principal Curve from the 3D points that are associated with the connected component.

Due to the dense boundary sampling, which is required for the Voronoi diagram, the skeleton graph consists of a large number of vertices. Since the 2D skeleton is merely a tool for analyzing the component shape, we apply a smoothing operation to facilitate further processing by reducing the

Algorithm 7.1 Pruning of spurious Voronoi edges with Potential Residual [Ogniewicz-1995]

```
1  procedure FILTEREDGES(G: skeleton graph with edge set E)
2      for all e ∈ E do
3          if e.isVoronoiEdge() then
4              (b_U, b_V) ← e.getGenerators()   ▷ Get boundary elements that generated this edge.
5              e.w ← dist^B(b_U, b_V) < minDistance
6          end if
7      end for
8      nbEdgesBefore ← 0
9      nbEdgesAfter ← |E|
10     while nbEdgesBefore ≠ nbEdgesAfter do
11         if only 2 vertices in G are leaves then        ▷ Graph consists of one connected path.
12             break
13         end if
14         nbEdgesBefore ← nbEdgesAfter
15         for all e ∈ E do
16             if e.w ∧ e.hasLeafVertex() then
17                 remove e from E
18             end if
19         end for
20         nbEdgesAfter ← |E|
21     end while
22     remove all vertices from G that have no incident edges
23 end procedure
```

number of skeleton nodes. Similar to Kégl and Krzyżak [Kégl-2002], we use a method proposed by Eu and Toussaint [Eu-1994] to smooth paths between two irregular vertices of the skeleton. A new path between the two end vertices is determined with fewer vertices than before. The deviation of the vertices that are removed is constrained to be no larger than Δs px. The result of the smoothing is a simplification of all paths in the skeleton graph. Clearly, skeleton smoothing is not essential for component segmentation as it does not change the topology of the skeleton. However, we found that it favorably influences processing in terms of performance and clarity when temporary processing results are evaluated.

After filtering and smoothing, the skeleton represents only significant component topology, except for a small number of possibly spurious edges. The next step is the attachment of component elements to skeleton nodes to prepare the component for segmentation.

7.2.2 Component Segmentation Based on 2D Skeleton

Each component element needs to be associated with its closest, adequate skeleton node to propagate the imminent segmentation of the 2D skeleton later amongst the 3D points. The minimum distance of an element to a skeleton edge e is the Euclidean distance

$$d_e(p) = \left\| \text{proj}_e(p) - p \right\| \tag{7.3}$$

between the 2D coordinates of the element p and its projection point $\text{proj}_e(p)$

$$\text{proj}_e(p) = \frac{(p-a)^T \cdot (b-a)}{(b-a)^T \cdot (b-a)} \cdot (b-a) \tag{7.4}$$

on the skeleton edge $e(a, b)$. In order to be a valid projection, the projection point $\text{proj}_e(p)$ has to be a point on the edge e: The corresponding line parameter t has to be in the interval $t \in (0..1)$. Otherwise, the point p is closer to one of the vertices a or b of the edge e.

For a skeleton vertex a, the minimum distance is the Euclidean distance between the element p and the 2D coordinates of skeleton vertex a

$$d_a(p) = \|a - p\| \tag{7.5}$$

However, assigning an element to the closest of all skeleton parts is not adequate. As demonstrated in figure 7.5, the skeleton node, which is closest to a point w.r.t. Euclidean distance in 2D might not be the proper skeleton branch it evidently belongs to. For this reason, it is important that assignments are determined under the constraint that boundary edges should not be intersected by the imaginary line that connects the considered point p with its respective skeleton node. Algorithm 7.2 summarizes the basic task. An element can only be attached to one skeleton node. If an element has the same distance to more than one skeleton node, a particular skeleton node is picked arbitrarily. For the experiments, this point location problem has been solved with a more sophisticated strategy, which is outlined in appendix A, to improve perfomance.

Finally, the skeleton is subdivided into a set of paths. Because the component elements are associated with skeleton nodes, segmentation of the skeleton graph implicitly partitions the elements as well.

First, the skeleton graph is contracted to an intermediary graph structure. Starting from a structure as in figure 7.6a, we build a condensed graph from the skeleton G_S to facilitate the path retrieval. All irregular vertices of the skeleton graph are also part of the condensed graph G_C. A sequence of edges between regular vertices, which connects two irregular vertices, is contracted to a single

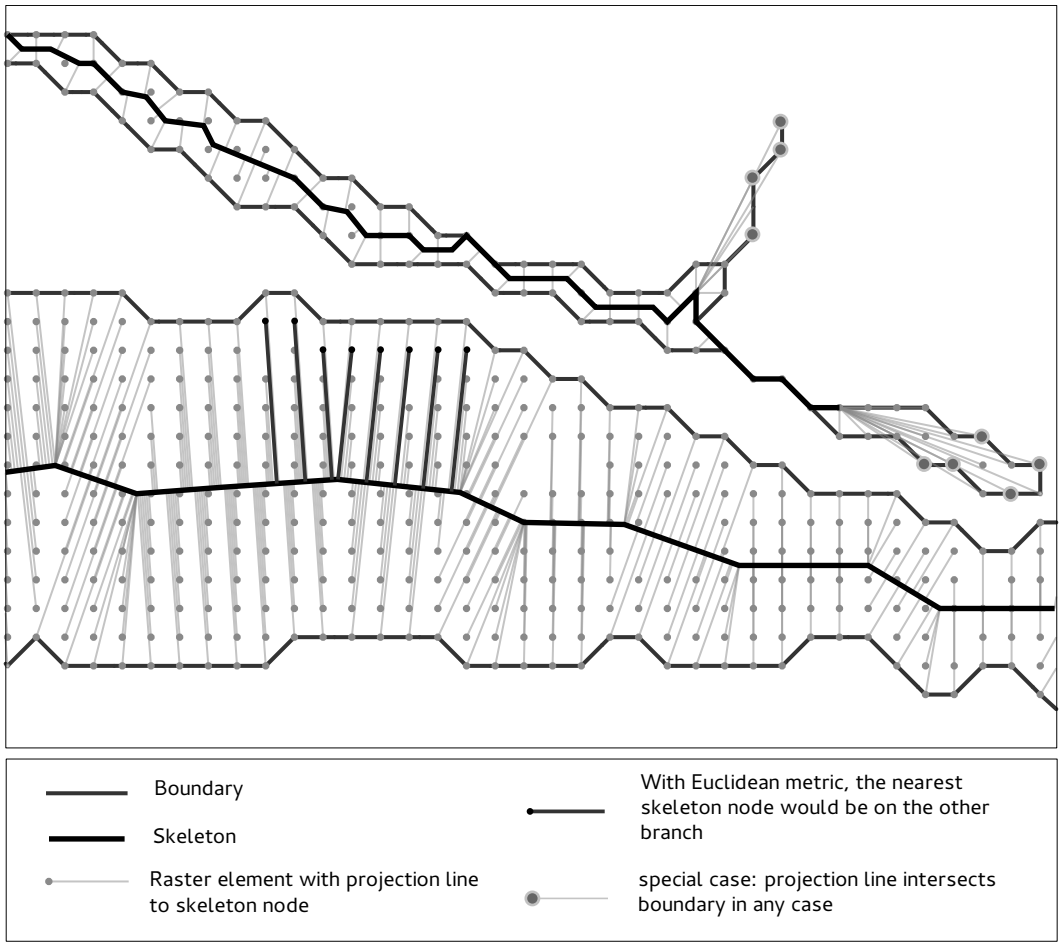


Figure 7.5: Points are attached to their closest skeleton node under the constraint that boundary edges should not be intersected if possible.

edge in G_C. Before edge insertion, the corresponding path between the two irregular vertices is tested whether it forms an arc: A path vertex that is adjacent to an irregular vertex is picked. If the distance from this vertex to all other path vertices increases monotonously, when they are tested in sequential order, then the path is represented by a single edge in G_C. Otherwise, the path is split into two parts at the vertex that is furthest from the considered vertex. The regular vertex is added to the condensed graph, as well as two edges representing both parts of the path in the skeleton graph. Furthermore, each edge in G_C is annotated with the properties N_e, and L_e: N_e is the number of points that are associated to skeleton nodes, which are represented by the considered edge in G_C; L_e is the length of the path in G_S, which is represented by the edge in G_C. The resulting condensed graph in comparison to the skeleton graph is depicted in figure 7.6b.

Second, segmentation is performed by retrieving paths from the condensed graph G_C. The graph is traversed in depth-first order starting at the root. Similar to Gorte [Gorte-2006] and Bucksch [Bucksch-2011], the root vertex is selected as the vertex with the largest y-coordinate considering the coordinate system of the 2D raster with the origin at top-left. At each vertex, we test whether any of the incident edges is an appropriate continuation of the edge that has been traversed before. Since the aim of segmentation is the computation of a Principal Curve from 3D point clusters, the characteristics of the PLA have to be taken into account during path retrieval. In other words, point clusters should have a rather uniform distribution of points along the unknown but discernible space curve in order to obtain a smooth polyline. If the point distribution is unbalanced, the ability of the polyline to describe the point set degrades respectively.

Algorithm 7.2 Basic description of point attachment

```
1  procedure ASSIGNPOINTS2SKELETONNODES(G_skel: skeleton graph, B: set of boundary segments, P: 2D point set)
2      for all p ∈ P do                                          ▷ Iterate over point set.
3          d_min ← ∞
4          n_min ← ∅
5          for all nodes n ∈ G_skel do                           ▷ Iterate over skeleton nodes.
6              L ← line segment of minimum length between p and n
7              if ‖L‖ < d_min then
8                  m ← 0
9                  for all boundary segments b ∈ B do            ▷ Iterate over boundary segments.
10                     if L intersects b then
11                         break
12                     else
13                         m ← m + 1
14                     end if
15                 end for
16                 if m = |B| then
17                     d_min ← ‖L‖
18                     n_min ← n
19                 end if
20             end if
21         end for
22         if n_min ≠ ∅ then
23             n_min.add(p)                                      ▷ Assign p to the determined skeleton node.
24         else                                                  ▷ All tested connection lines intersect the boundary.
25             find a path from p to a skeleton node for instance using the raster
26             adjacency graph (see section 6.3.2) with breadth-first traversal
27         end if
28     end for
29 end procedure
```

For this reason, we evaluate the ratio R for each edge e

$$R(e) = \frac{N_e}{L_e} \tag{7.6}$$

that describes how many points are projected onto the skeleton path in the length of a unit measure. The ratio of a candidate edge e_A is compared to the ratio of the last traversed edge e_C by calculating

$$w(e_A, e_C) = \frac{\min(R(e_C), R(e_A))}{\max(R(e_C), R(e_A))} \tag{7.7}$$

The candidate edge e_A that has the highest weight is selected and tested whether the weight exceeds a predefined threshold ϵ_{weight}. If $w(e_A, e_C) > \epsilon_{weight}$ then the candidate edge e_A is checked whether the already identified path that is associated with the previously traversed edges would form an arc if e_A, i.e. its skeleton path is appended. In this case, the candidate edge is discarded and the current path is terminated. If $w(e_A, e_C) \leq \epsilon_{weight}$ then the candidate edge is immediately discarded because the respective ratios differ by more than $(1 - \epsilon_{weight}) \cdot 100\%$. In this case, the current path terminates and new one is started. In the best case, a path can be successfully traversed until a leaf vertex of the condensed graph is encountered. An example of the resulting path partitioning is given in figure 7.6c. Irregular vertices belong to all their incident paths at the same time.

If the skeleton graph contains cycles, the condensed graph will contain the same cycles as well. However, the condition that a path must not form an arc suppresses the retrieval of a path that is a cycle. At present, cyclic structures are not addressed further in processing.

After path retrieval, the point clusters that are subjected to the Principal Curve computation can be collected. Each retrieved path of the condensed graph represents an ordered set of nodes of the skeleton graph. Since point subsets are associated with each skeleton node, the point cluster of a particular path can be obtained as the union of the point sets that belong to the respective skeleton nodes as depicted in figure 7.6d. The relation between 2D raster coordinates and 3D coordinates from TLS data is known as explained in section 6.2. Therefore, the result of the operation is a segmentation of the component into clusters of 3D points.

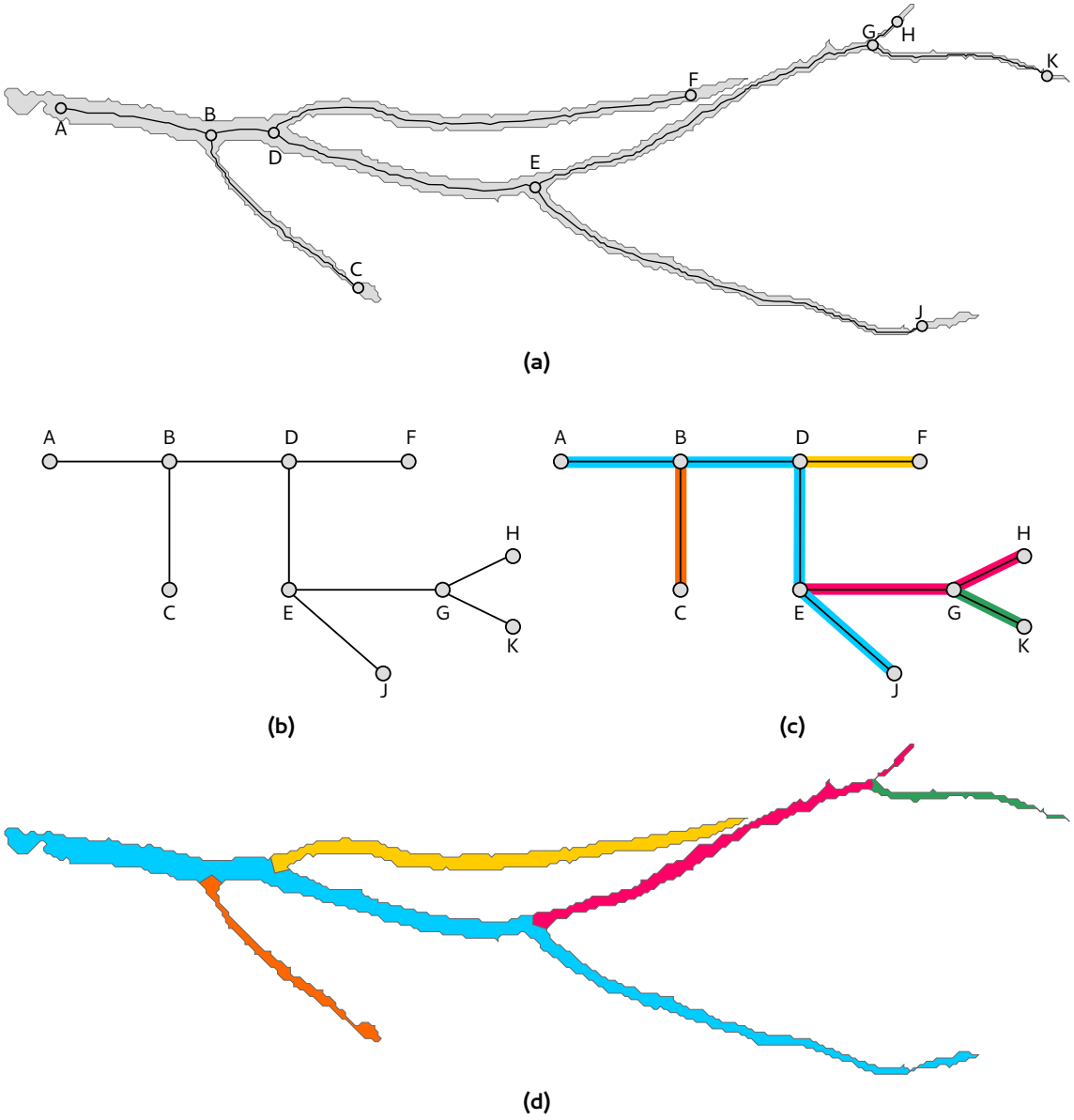


Figure 7.6: Segmentation of the connected component on the basis of the 2D skeleton: (a) Smoothed skeleton G_S that is the starting point. (b) The topology of the skeleton is represented as the condensed graph G_C. (c) Here, the root vertex for path retrieval was A because the illustration is rotated by 90°. Retrieved paths are mapped to colors. (d) The paths partioning is propagated to all elements of the connected component.

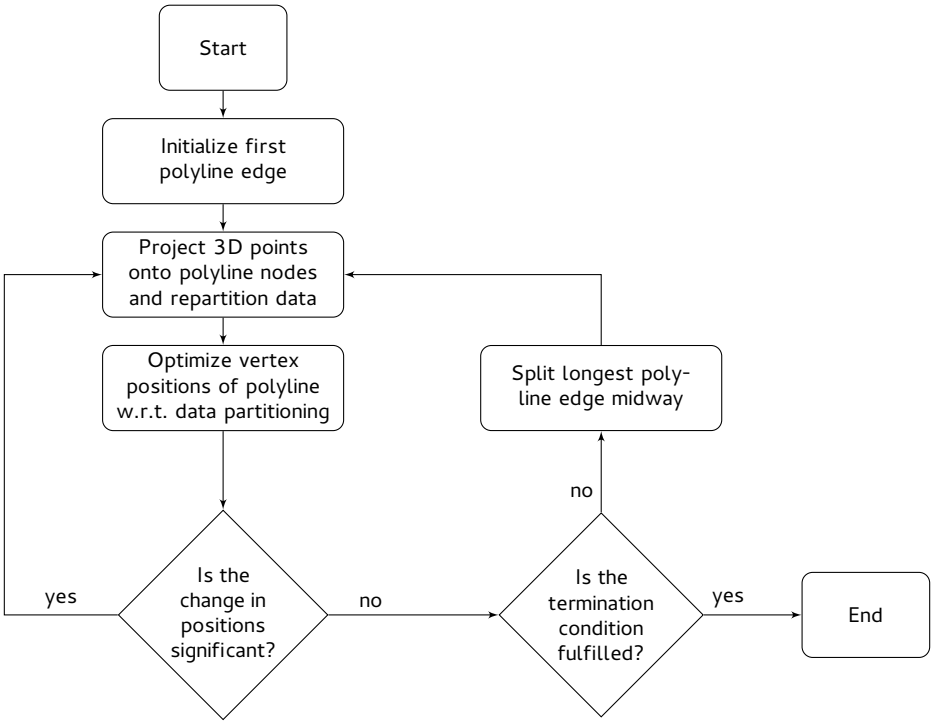


Figure 7.7: Outline of the Polygonal Line Algorithm (PLA, [Kégl-1999])

7.2.3 Retrieval of Principal Curves from 3D Point Subsets

Before a 3D point cluster is subjected to the PLA, data normalization is performed as suggested by Hartley and Zisserman [Hartley-2003]. The 3D point cluster D is transformed isotropically with the homogenous matrix T

$$T = \begin{bmatrix} \frac{\sqrt{2}}{m} & 0 & 0 & 0 & -\sqrt{2} \cdot \frac{\mu_X}{m} \\ 0 & \frac{\sqrt{2}}{m} & 0 & 0 & -\sqrt{2} \cdot \frac{\mu_Y}{m} \\ 0 & 0 & \frac{\sqrt{2}}{m} & 0 & -\sqrt{2} \cdot \frac{\mu_Z}{m} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.8}$$

with μ as mean and m as mean value of the standard deviation of $D \in \mathbf{R}^3$ [MVG-2004].

Subsequently, the PLA, which is outlined in figure 7.7, is applied to the normalized point cluster. First, an initial straight line estimate is generated as the eigenvector that corresponds to the largest eigenvalue, which is calculated by a Principal Component Analysis [Jolliffe-2002]. Afterwards, all data points are projected onto the polyline estimate and partitioned according to their closest polyline node. An optimization follows that iteratively moves polyline vertices along their approximated gradient vectors in order to minimize a quality measure of the polyline. This measure is calculated from the mean square distance error of each vertex to its associated point set. The optimization routine terminates if the change in polyline quality between two iterations is lower than a predefined threshold. Then, the polyline edge with the highest number of data points is subdivided by adding a new polyline vertex, which splits the edge midway. Data projection, optimization, and insertion of a new vertex are repeated iteratively until a termination condition is fulfilled. The termination condition takes the current number of polyline edges, the overall smoothness of the polyline relative to extend and cardinality of the data set into account. As illustrated in figure 7.8, the polyline becomes smoother in each iteration and reflects the distribution of the data points more and more.

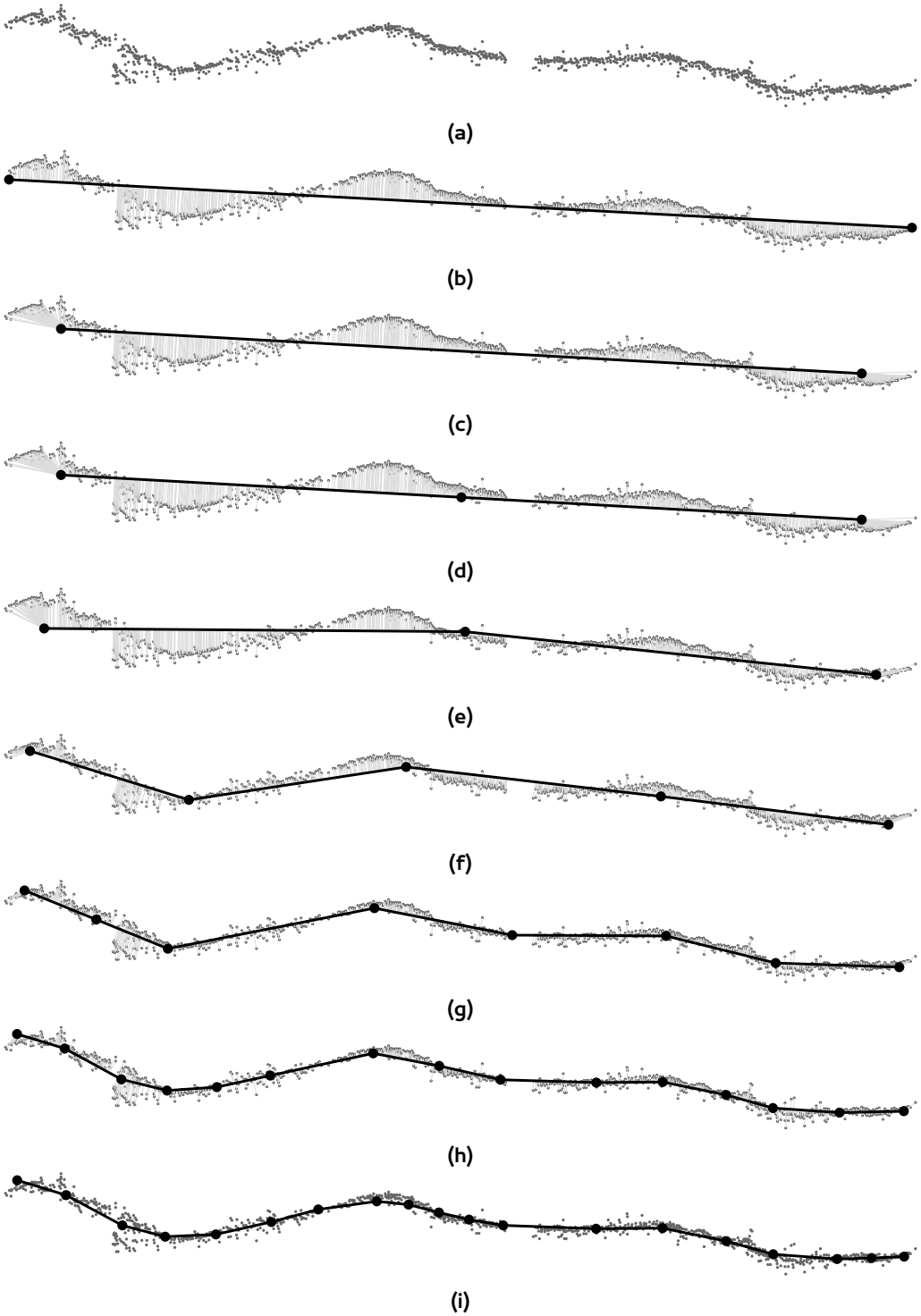


Figure 7.8: The polyline that is initialized from (a) the 3D point set as (b) a straight line segment is approximated more closely in each iteration. b) The points are projected onto the polyline. (c) The optimization draws the polyline vertices inwards. (d) The edge is split midway. Subfigures (e)-(h) depict the points and polyline after a number of iterations. The final result is shown in (i).

Figure 7.9: The segmentation from figure 7.6d is colormapped to the 3D point set. The 3D polylines that are the result of the PLA on the point groups are shown in black.

The resulting 3D polyline is then transformed back to the original coordinate frame by multiplying its 3D coordinates of vertices with the inverse of the transformation matrix T

$$T^{-1} = \begin{bmatrix} \frac{m}{\sqrt{2}} & 0 & 0 & 0 & \mu_X \\ 0 & \frac{m}{\sqrt{2}} & 0 & 0 & \mu_Y \\ 0 & 0 & \frac{m}{\sqrt{2}} & 0 & \mu_Z \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.9}$$

that is obtained from the original data set D. Figure 7.9 demonstrates the resulting set of 3D polylines that corresponds to the segmented connected component in figure 7.6d.

As a result of the path retrieval procedure, every path has a parent path, except for paths that contain the root vertex. The parent of a particular path is the one that terminates at the very vertex in G_C, where the considered path starts. Consequently, the 3D polyline has a parent polyline, too. Though, joining of the set of resulting polylines to form a connected graph that represents the component's skeleton is a task that belongs to future development. At present, the skeletal structure of a particular connected component is retrieved as a set of disjoint polylines.

7.3 Experiment Setup and Results

All three data sets B, K, and E were used in the experiments. The presented method is implemented in C++ utilizing Eigen [Eigen] and Boost [Boost] libraries, which also provide the computation of a Voronoi diagram. The Delaunay triangulation is constructed using Triangle [Shewchuk-1996]. The PLA that was available in [PLA-1999] was ported to C++. The point attachment routine as summarized in algorithm 7.2 is a naïve formulation of the task. Therefore, a more sophisticated assignment strategy, which is explained in appendix A, was implemented to accomplish the task. Again, the experiments were done on a dual-core machine (4 GB RAM, Linux, 64 bit).

The input data was provided in form of IAL files. Similar to the experiments in section 6.5, only points located within a cylindrical bounding volume at the tree position are considered for tree data from data set B, and K. In addition, all input data was manipulated with the objective to remove the ground surface. Restriction of the data is justified by the fact that patches of ground surface or understory vegetation are not of interest in the context of this thesis. Therefore, the data was reduced to measurements, which are located above the XY-parallel plane at $Z = p^Z_{scanner} + 5\,\text{m}$. For all scans in chapter 4, the TLS instrument was positioned at least in 1.60 m above ground level. With the additional offset, it is ensured that the ground surface is eliminated even if the terrain is slightly sloped. As a consequence, tree trunks are pruned to a varying degree, which is neglected in the experiments. Since our main focus is on the retrieval of branching structure of the tree crown, we refer to methods that have been discussed in section 2.3 to model tree trunks from TLS data.

Furthermore, the experiments have been limited to compute only connected components between 1,000-100,000 raster elements in size for each input data set. Smaller components are omitted because the probability that a component is an apparently uninterpretable noise artifact is inversely proportional to its number of elements. Larger components are omitted because most often they represent considerable patches of trunk surface. Especially if the tree was close to the scanner, the trunk surface is amply sampled by points. At present, the performance of our method drastically decreases if huge patches of trunk surface are processed. As mentioned previously, there are more suitable methods for trunk reconstruction in the literature.

The number of elements that belong to a component influences the overall processing time linearly. The point attachment procedure currently denotes the computational bottleneck of the implementation. In general, point location tests are commonly rather costly. Moreover, the iterative repartitioning and optimization, which is performed by the PLA, is also chiefly governed by the number of points. However, the overall time that is necessary to retrieve the spatial structure of a moderately large connected component, which represents branching structure, is in the magnitude of seconds up to a few minutes. The maximal time that elapsed for a component in the experiments was 2.7 min. The average runtime is below 3 s for the 2543 tested components. Results of the experiments are demonstrated in figure 7.10, 7.11, 7.12, and 7.13.

There are only few parameters that control the processing. Again, the threshold $\epsilon_{CCL} = 0.05\,\text{m}$ has been used for the CCL procedure, which creates the basis for the subsequent structure retrieval. The boundary tracing itself does not require any parameters. But we test all obtained boundaries of a component to filter out particularly small polygons, which are most likely data gaps. A boundary sequence has to consist of more than 4 elements and its area on the 2D raster has to be larger than $t_A = 5$ px to be included in the computation of the Voronoi diagram. After linking the sub graphs, the Voronoi edges of the initial skeleton estimate are filtered to remove spurious edges. The Potential Residual [Ogniewicz-1995] of a Voronoi edge has to be higher than $t_{PR} = 40$ px. The threshold ϵ_{weight} that controls the continuation of paths during the path retrieval procedure has been set to $\epsilon_{weight} = 0.6$. In other words, the ratios $R(e_A)$ and $R(e_B)$ should not differ by more than 40% relative to the largest of both values. The threshold Δs that is used in the smoothing operation has been set to $\Delta s = 0.25$ px. As a result, only skeleton vertices with incident edges

that form an almost straight angle are removed by the smoothing. The set of parameters has been chosen after empirical testing and kept fixed for all experiments. The parameters that control the PLA have been set to the same values that were preset in [PLA-1999]. Though, the PLA has only been applied to point clusters that consist of more than 10 points.

The output of the processing is a set of 3D polylines for each detected connected component of the input data set, which has been compliant with the predefined size constraint.

7.4 Discussion

The experiment results show that the method retrieves the spatial structure with higher reproduction accuracy than the first method (see chapter 5). Intertwined crowns or the definition of an appropriate bounding volume are no issues for this method. In contrast to the voxel space approach, a lot more details and smaller structures can be adequately reconstructed because the method operates directly on the scan point level. On average, the time and memory resources that are required for processing are modest, which is favorable.

The obvious drawback is that only partial skeletons of the entire tree in the scan are recovered. However, possibilities to develop the presented method further to overcome the current limitations are addressed in section 7.5. In the following, we concentrate on the properties of the subprocedures and discuss their interdependence within the processing scheme.

The CCL procedure has been discussed previously in section 6.3.1. For the experiments, we have restricted the data to above ground measurements prior to CCL. Another key point in the data manipulation is that a connected component of a tree, which stood close to the scanner, represents a large part of the trunk and a considerable patch of ground surface. In this form, it is an inappropriate input for the method. It is important to realize that the method will produce a result for any input component. But if the spatial structure cannot be adequately summarized by a set of space curves in the first place, then the output will certainly be unsatisfying and not reflect the component structure as expected.

Boundary information is fundamental for the approach. Besides the main boundary, all depth discontinuities, which are embedded in the connected component, are properly traced as explained in section 6.3.3. Evidently, larger components have commonly a higher number of holes caused by missing data than smaller components. The majority of these data gaps does not represent relevant shape features of the object that was scanned. For this reason, we remove hole boundaries by testing against simple thresholds regarding the boundary size and polygon area. But the selection of suitable parameters is difficult due to the variety in size of components and holes. If hole boundaries that contribute relevant shape information are removed by filtering, the reproduction accuracy of the resulting skeleton decreases. If irrelevant holes are included in the method, the initial skeleton will reflect them as cycles, which also eventually degrade the final result. Clearly, this problem requires special attention to enhance the method further. Boundaries and their neighboring elements could be thoroughly analyzed to distinguish relevant from irrelevant holes. Moreover, the initial skeleton that is obtained on the basis of the boundary information could be restructured to compensate such spurious topology.

A skeleton of the 2D component is efficiently determined via the Voronoi diagram. The high number of perturbations in component boundaries leads to an equally high number of spurious edges. This problem can be approached from two sides: If the boundary is smoothed before the Voronoi diagram is calculated, less spurious edges would emerge. Similarly, spurious edges can be removed by filtering after calculating the Voronoi diagram as presently done in the method. Then, the problem is again: How to determine whether a particular Voronoi edge is generated by

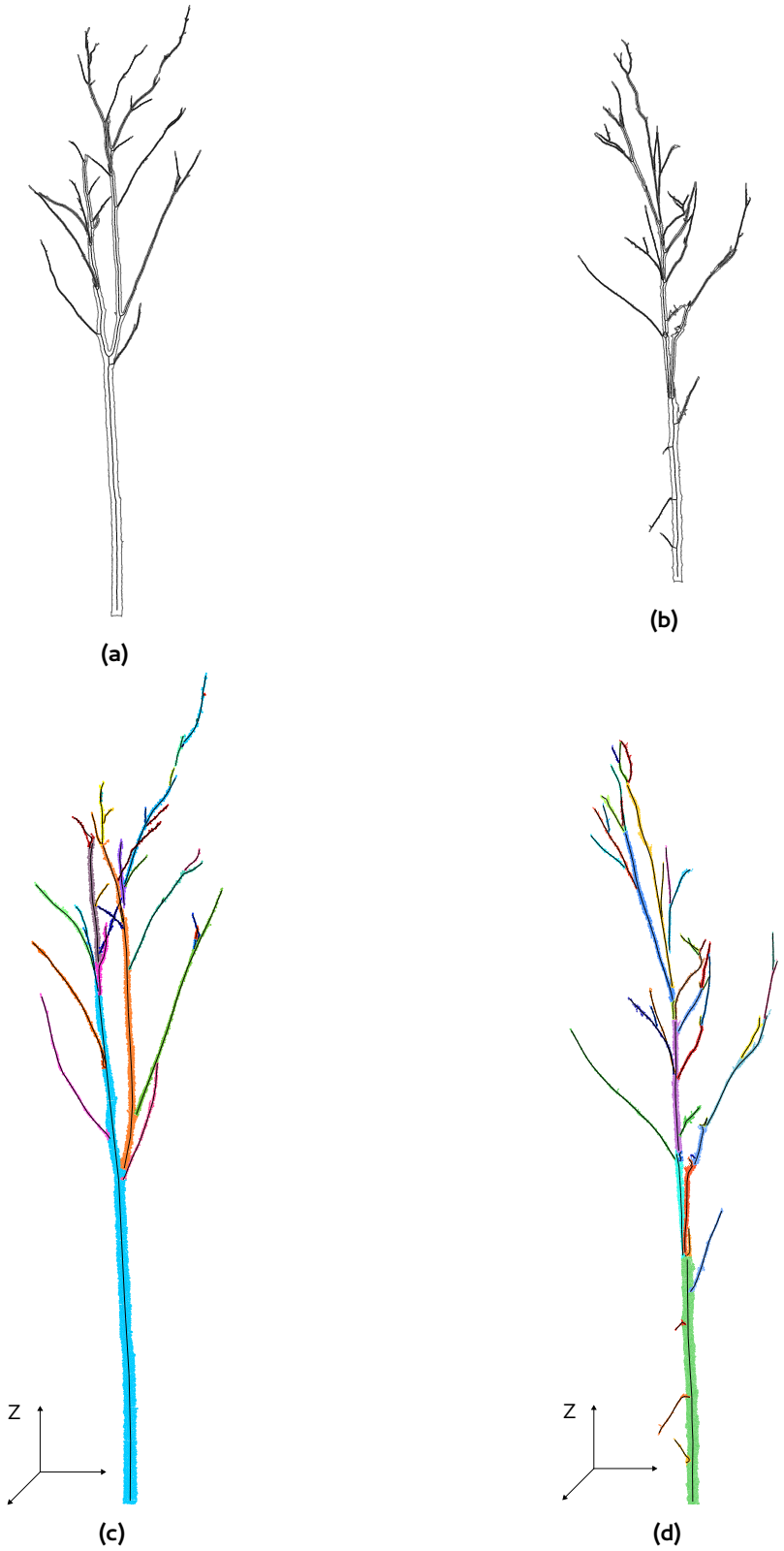


Figure 7.10: Results of the experiments on data set B. Subfigure (a) and (b) show the boundaries of the connected component in gray and the 2D skeleton in black. Subfigures (c) and (d) show the 3D point sets and retrieved skeletal structures that correspond to (a) and (b). Each colormapped point group represents a subset that generated a polyline via PLA. The 3D polylines are drawn as overlay onto the corresponding point subsets.

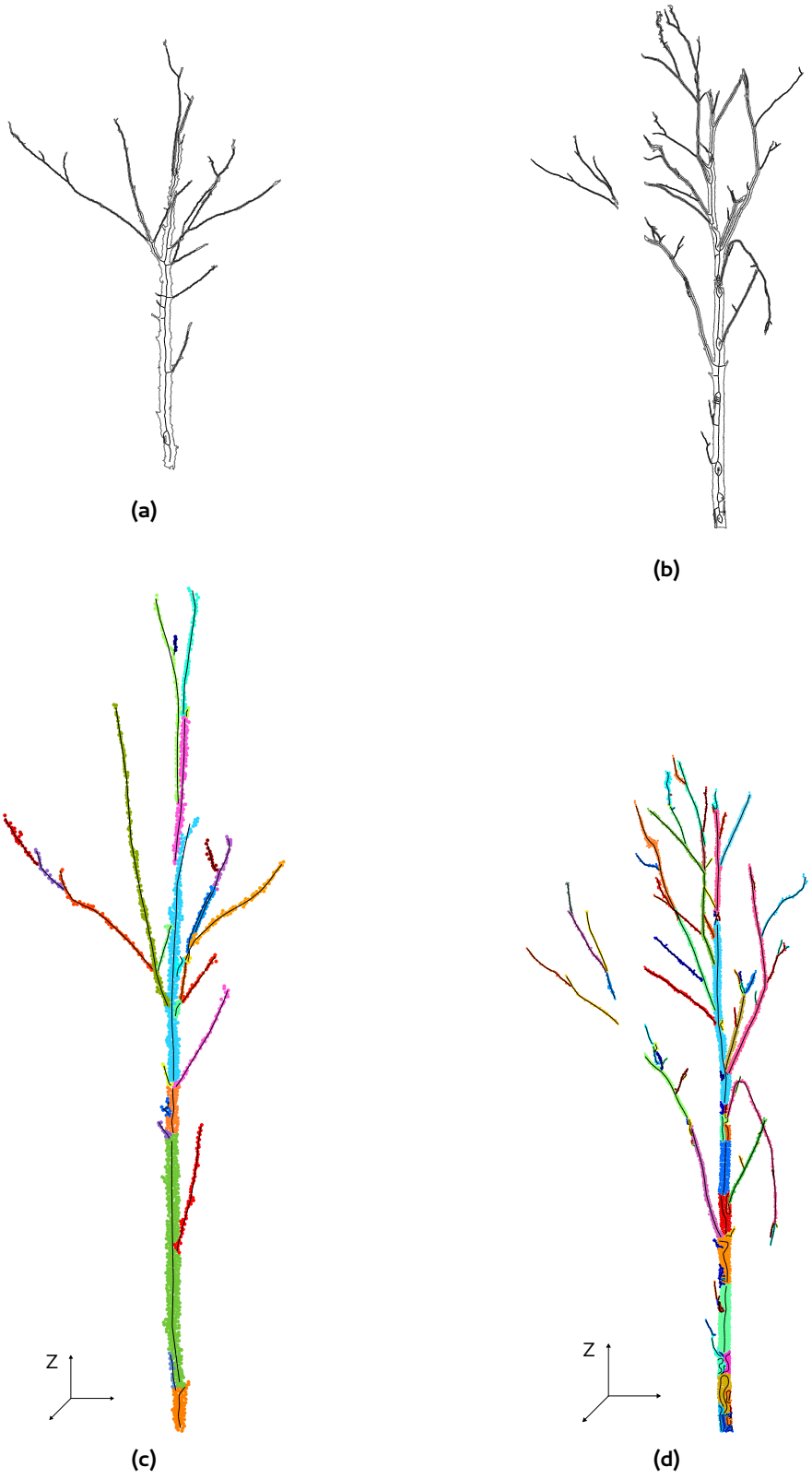


Figure 7.11: Results of the experiments on data set B. Subfigure (a) and (b) show the boundaries of the connected component in gray and the 2D skeleton in black. Subfigures (c) and (d) show the 3D point sets and retrieved skeletal structures that correspond to (a) and (b). Each colormapped point group represents a subset that generated a polyline via PLA as in figure 7.10

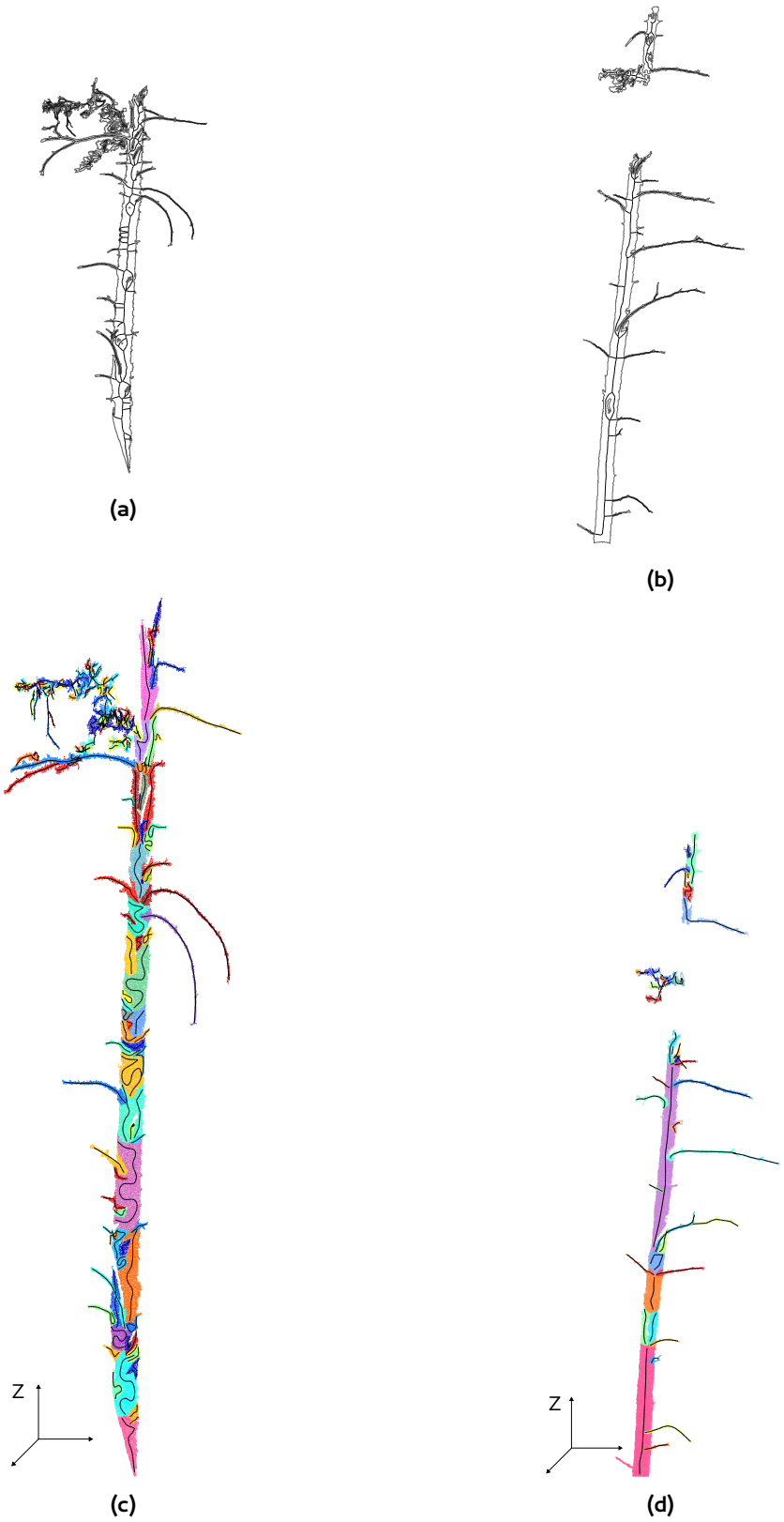


Figure 7.12: Results of the experiments on data set P. Subfigure (a) and (b) show the boundaries of the connected component in gray and the 2D skeleton in black. Subfigures (c) and (d) show the 3D point sets and retrieved skeletal structures that correspond to (a) and (b). Each colormapped point group represents a subset that generated a polyline via PLA. The 3D polylines are drawn as overlay onto the corresponding point subsets.

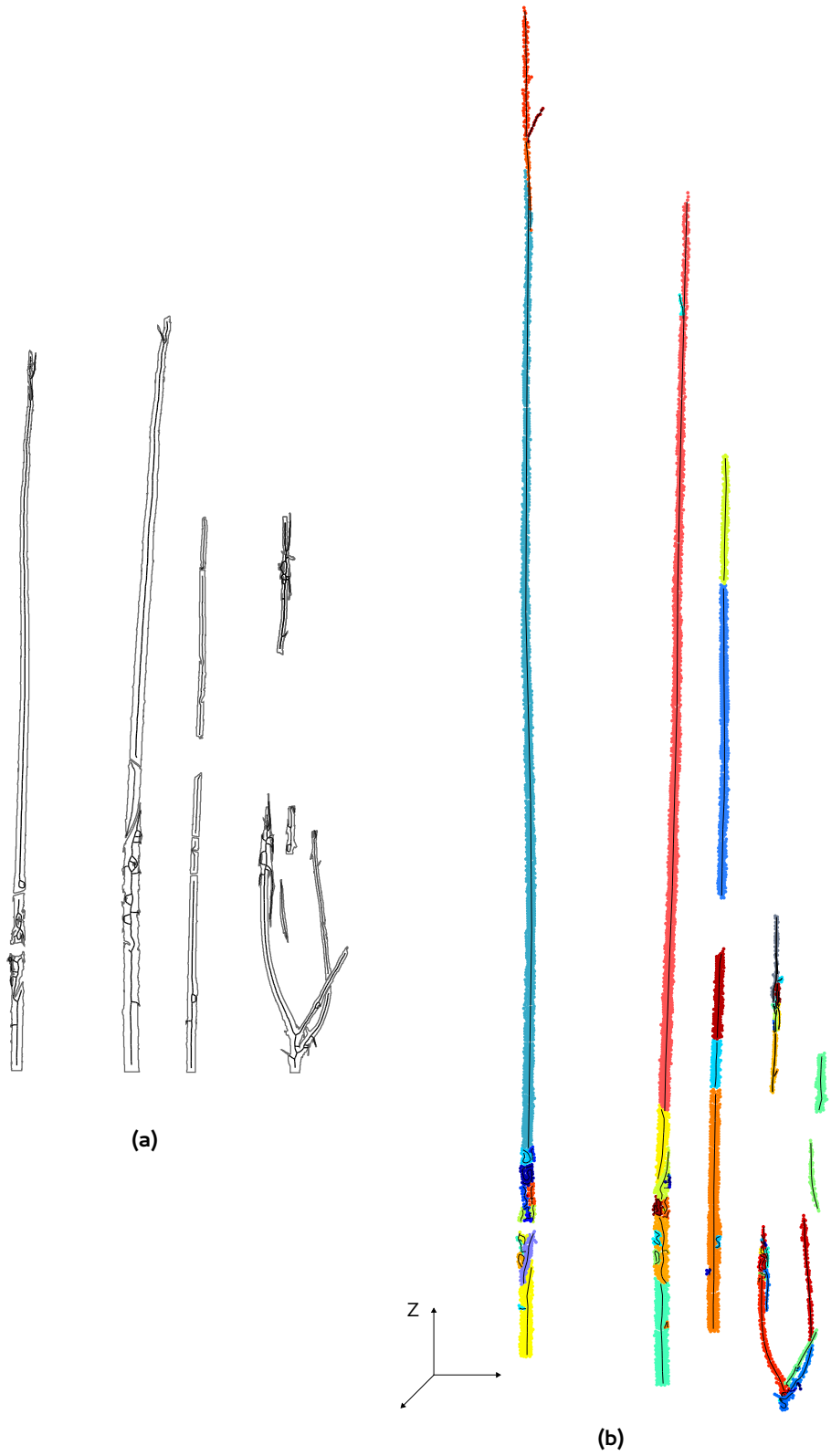


Figure 7.13: Results of the experiments on data set E. Subfigure (a) shows the boundaries of the connected component in gray and the 2D skeleton in black. Subfigure (b) shows the 3D point sets and retrieved skeletal structures that correspond to (a). Each colormapped point group represents a subset that generated a polyline via PLA. The 3D polylines are drawn as overlay onto the corresponding point subsets.

a relevant shape feature or not? Currently, the question is answered with the help of the Potential Residual by Ogniewicz and Kübler [Ogniewicz-1995]. While maintaining connectivity, we suppress Voronoi edges generated by perturbations that are described by a boundary section, which is less than $t_{PR} = 40$ px in length. In most cases, the threshold produces appropriate results in the experiments. However, the arbitrary definition of the parameter is clearly not of universal validity and the filtering consequently removes to many or too few Voronoi edges in some cases. In this context, the straight cutting edge of components from trunks, which have emerged as a result of the pruning prior to CCL, influences the skeleton estimate favorably.

Before filtering Voronoi edges, the skeleton sub graphs are linked to a single graph. If this is not performed in advance, smaller sub graphs may be entirely eliminated because of the filtering. For the linking procedure, a constrained confirming Delaunay triangulation is constructed and used as search space. Nodes of skeleton sub graphs are annotated accordingly in the interim support structure. Then, breadth-first search is started at a particular skeleton vertex. The search terminates when the shallowest goal node is found, i.e. as soon as a vertex, which belongs to a different sub graph, is encountered. The length of the search path refers to the number of vertices that are traversed until the goal vertex is reached. As a result, the obtained path is not the shortest path w.r.t. the Euclidean metric, but the path from start to goal with the least nodes. For this reason, it is important that the Delaunay triangulation is not only constraint to include the skeleton, boundary, and hole nodes, but also to be truly confirming to the Delaunay condition. Due to the Delaunay criterion, the triangles of the resulting mesh are compact. Consequently, the search path has not necessarily the shortest Euclidean length, but it is normally a suitable approximation of it, which is sufficient for solving the task. In the experiments, our linking strategy created connections between disjoint sub graphs as expected. But actually the linking procedure does not rely on this particular formulation and could be replaced with an equally adequate mechanism.

The sequence of procedures to obtain and refine a 2D skeleton from a particular connected component is intricate and strongly interdepending. The resulting 2D skeleton lays the foundation for the following segmentation. In fact, truthful as well as spurious topology of the 2D skeleton is propagated to the set of 3D polylines. Therefore, the quality of it is of essential importance to the final result of processing.

The segmentation of the connected component into point clusters is subdivided in point attachment and path retrieval. The basic algorithm to attach points to their nearest, appropriate skeleton node with respect to the boundary limits has been summarized in algorithm 7.2. For the experiments, a sophisticated solution had been implemented, which is explained in appendix A.

It has to be kept in mind that the component boundary, which includes inner rifts, represents depth discontinuities. In other words, adjacent elements of the component are close in 2D, but possibly far away in 3D space. Therefore, it results in false point segmentation if the imaginary line that connects a particular point to its respective skeleton node intersects the boundary. However, component elements that are at the same time also boundary elements need to be specially handled. In case of appendices, bridges or boundary polygons, which do not contain skeleton nodes, the component elements are attached simply to the closest skeleton node. Naturally, the resulting imaginary connection line most likely intersects the boundary. In general, the number of such occurrences is low in comparison to the total number of component elements. In the experiments, we have not noticed a significant influence on the segmentation. For this reason, this trivial solution suffices for the few special cases, which are not addressed further. For all other component elements, the imaginary connection line to the skeleton node must not intersect the boundary.

Following point attachment, the actual segmentation of the component is performed in the path retrieval procedure. The skeleton graph is contracted to a graph, which mainly represents its topology. The geometry is only queried to test whether a candidate edge would form an arc when appended to the current path. The test was integrated because the PLA cannot model this partic-

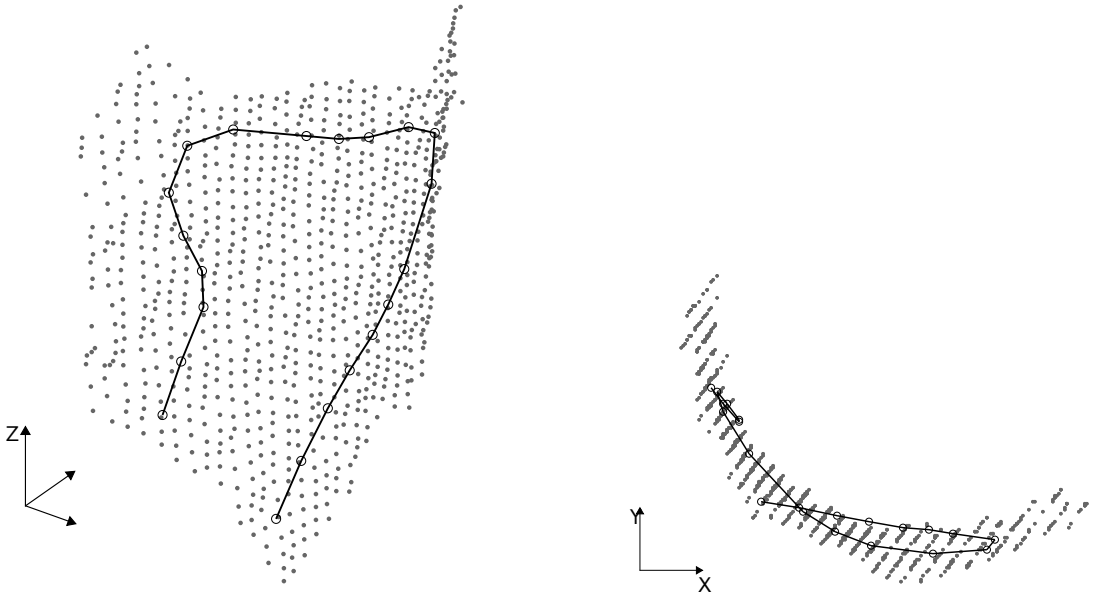
ular point configuration. In the 2D skeleton, it is simple to recognize such a case and ensure that it is represented by two separate polylines. On the basis of preliminary testing, we found that splitting an arc into two parts is an appropriate solution, since branches are not growing exceedingly twisted in general.

Similarly, the continuity criteria was designed to obtain point clusters that are especially suited as input for the PLA. The geometry of the skeleton is not considered. Therefore, the path retrieval does not rely on any predefinition or limitation of the growing direction. The continuity of paths at graph vertices is based on the assumption that smaller branches fork off into a different direction from the stronger one, which keeps its previous growing direction also after the ramification. Furthermore, we assume that the number of sample points changes only gradually on a particular surface patch. In addition, abrupt changes in surface continuity are not to be expected on a tree. In other words, one can observe that a smaller branch is sampled by less scan points than its parent branch because the latter faced the scanner with a larger surface area. Consequently, it leads to results of higher quality from the PLA if the two parts of the stronger branch before and after the ramification are merged in the same point cluster. The smaller branch, which forks off, is an inadequate choice to continue the stronger branch after the ramification because it has fewer points. An inhomogeneous point distribution is reflected in the polyline that is computed by the PLA, but this is often undesired. For this reason, we based the path retrieval on the proposed continuity criterion, which takes the path length and the number of attached points into account with the objective to generate clusters of homogeneous point distribution. Clearly, a small number of branches that are represented with a high level of detail is more favorable than a large set of polylines of insufficient quality.

At the same time, the path retrieval reacts sensitive to spurious skeleton edges exactly because of the continuity criterion. After the point attachment, a set of points is associated with the spurious edge, which should have been assigned to the nearby true skeleton nodes. Therefore, the value of the ratio R of the apparently true skeleton edges decreases because of the spurious point attachment to another skeleton node. The difference between the ratios of successive edges then does not exceed the predefined threshold. As a consequence, the current path is terminated and a new path is started. The component shape is represented by more polylines than actually necessary. Most importantly, the partitioning into paths has an immediate impact on the point distribution of a particular cluster, which chiefly governs the polyline that is eventually generated from it.

Generally speaking, the segmentation of a component into separate clusters of 3D points depends strongly on the previous processing steps. The point distribution of the obtained clusters can exhibit a great diversity. But not every point cluster is an appropriate input to the PLA, which tries to summarize a given set of points as a polyline. If a point set has been sampled on the surface of a branch and the invisible space curve, which is traced by the points, is clearly perceptible by a human, the PLA will generate an appropriate polyline. But if the point set really represents nothing but a patch of surface, then a polyline is definitely not the proper geometric primitive to summarize this shape. To put it simple: If a human cannot perceive the invisible space curve in a point set, then the PLA will yield unexpected, unintended results. For instance, the point set in figure 7.14 is summarized by the PLA as a meandering polyline embedded in the point set.

In the context of the proposed method, the difficulty lies in the fact that all scan points from TLS are surface points. Therefore, clusters represent exclusively patches of object surface. Solely the extent and distribution of a particular point cluster decides whether it traces an invisible space curve or not. Unfortunately, we have no possibility to test if a point cluster is a suitable input for the PLA. Clearly, retrieving information about the distribution of points is the very reason why the PLA is applied at all. But also the evaluation of the generated polylines poses a serious challenge. Currently, the PLA as available from [PLA-1999] is integrated as a kind of black box. Two parameters – MSE (mean of squared distances) and penalty – are delivered that inform about the state of the computed polyline.



(a) front view **(b)** top view

Figure 7.14: A sample representing a patch of surface results in a meandering polyline from the PLA.

The MSE is the mean value of the squared distance of all points to their associated polyline node. However, this measure cannot be used to distinguish apparently *good* polylines (e.g. figure 7.9) from *bad* ones that are spiraling or meandering as in figure 7.14. The aim of the optimization routine is to minimize the distances. For this very reason, the polyline assumes the meandering shape for point clusters that represent surfaces patches. But the degree to which the distances are minimized or not does not offer any information regarding the polyline shape.

The penalty of a polyline refers to the average angle between incident edges at vertices. If the polyline is rather smooth, the penalty is low. However, it does not necessarily indicate that the polyline shape is equal to a straight line. The penalty also depends on the number of polyline edges. If an arc of high curvature is represented by a high number of short polyline edges, the total penalty may also be rather low. Similarly, a higher penalty value is not an indicator that the polyline is meandering in a sample of surface area instead of tracing an invisible space curve. Commonly, branches do not grow as straight sticks, but bend and twist to some degree in possibly various spots. Naturally, the penalty would reflect this with a higher value, which would be totally appropriate in that case.

Given these points, we cannot assess a polyline quantitatively. Therefore, we rely on visual examination in order to examine the set of polylines at present. Yet, a Principal Curve as a polyline is an adequate tool to summarize a suitable point cluster, because it is a piecewise sequence of principal components with respect to an imposed subdivision of the point set. Therefore, the polyline is always centered in the given point set.

Beside Kégl [Kégl-1999], several other studies have investigated the notion of Principal Curves. Since the experiments have shown that Principal Curves are a valuable tool for retrieving skeletal structures from TLS data, it is naturally of interest to evaluate whether the proposed method can be enhanced by an alternative approach to Principal Curves. Further experiments with different algorithms that generate Principal Curves from point sets, but also extensive testing to find more suitable parameter values would certainly be beneficial.

Another crucial issue came to light during preliminary testing with the PLA: Data normalization appears to be of utmost importance for processing TLS data. A 3D point is usually represented

as floating point number with the IEEE 754-2008 standard [IEEE754-2008]. In contrast to integers, the spacing between floating point numbers is not homogeneous as pointed out for instance by Isenburg [Isenburg-2013]. In fact, a spot of high precision is located around zero, whereas the spacing between successive representable numbers becomes greater the further they are away from zero. In other words, a floating point number may have a large figure before the comma, but then only a very limited number of decimal places can be represented. In contrast, if a number is close to zero, it can be represented with many decimal places. In the scope of processing TLS data, this issue becomes relevant if small changes should be applied to point coordinates. For instance, the gradient vectors that are computed in the optimization routine of the PLA are rather small in length. The point coordinates are considerably large because of the co-registered multiscan. As a result, the addition of the gradient vectors to the point coordinates has no effect because the change in decimal places cannot be accurately represented. Consequently, the optimization routine functions properly but to no avail. On the contrary, since only decimal places right after the comma have any effect, the resulting polyline may even exhibit a zig-zag shape.

For this reason, we integrated a data normalization step prior to applying the PLA in order to prepare the coordinate frame of the points with respect to the targeted algorithm as suggested by Hartley and Zisserman [Hartley-2003]. By moving the point cluster to the origin and scaling it to a maximal extent of $\sqrt{(2)}$ along a coordinate axis, the point set is located at the aforementioned spot of high precision. In this way it is at least ensured that this issue has only a comparably tiny influence on the resulting polylines. By and large, issues pertaining to the digital representation of numbers as for instance the particularities of floating point arithmetic should always be kept in mind when processing laserscanning data.

Summing up, the proposed method constitutes a promising approach to retrieve skeletal structures from TLS data representing trees. The key idea is motivated chiefly geometrically with respect to the Medial Axis concept, which is favorable. The exploitation of the relationship between 2D raster coordinates and 3D point coordinates is clearly a vital aspect in the processing of TLS data. The scheme of sub procedures that is proposed to implement the key idea in practice is admittedly intricate. The processing is controlled by only few parameters, which is beneficial. However, the set of parameters has a considerable influence on the output of the method. Yet, the same parameter values have been used for all conducted experiments and mostly yield satisfying results w.r.t. the particular input data. The method is not limited to a particular tree species. Naturally, a higher number of satisfying results is obtained from leafless deciduous trees than coniferous ones. But the main reason for this behavior is the foliage of coniferous trees, which cannot be properly represented by space curves. However, single branches as for instance demonstrated in figure 7.12 can be successfully retrieved from the Scots Pine as well.

Another key point is that the method can be applied immediately to data of a single scan. Tree detection is not a mandatory pre-processing step, though helpful to evaluate the results. As a consequence, the method could also be tuned to detect trees.

As mentioned before, the apparent weakness of the proposed approach is the fact that only partial skeletons are generated from single scans. As a result, there is still a long way to go until a full skeleton can be retrieved automatically. In fact, the skeletal structures have to be interpreted as building bricks, which still need to be assembled to the true skeleton of the tree. This process bears the prospect of compensating wind effects, which are very often in TLS data of forest scenes, as well as occlusions.

7.5 Future Work

Beside the improvements that are clearly pending for the presented method, there are several tasks to solve in order to produce a full skeleton of a tree. The skeletal structures of a tree in a single scan need to be collected and combined if possible. For this task, the Voronoi diagram may be of use again to recognize neighborhood relations between connected components in the 2D raster. Naturally, the assembly of results from individual trees of a multiscan is the essential task to generate full skeletons of trees. As pointed out by Eysn et al. [Eysn-2013], this could lead to a more robust retrieval scheme for tree skeletons than existing approaches in the literature. In general, plenty of extensions and enhancements are imaginable for the proposed method, which is clearly a vital contribution in the scope of retrieving skeletal structures of trees from TLS data.

8 CONCLUSION

Information about the spatial tree structure is important for forest science to gain insights about the processes and interrelationships within the forest ecosystem. The rapid and contactless sampling of object surfaces as a 3D point cloud via TLS is a sensible choice to obtain geometry data of trees, which can assume an amazing variety of shapes. For this reason, the retrieval of skeletal structures from TLS data is also an ill-defined problem: Neither the starting point, nor the expected solution has a clear definition. Hence, the task has to be approached in a pragmatic way to develop working solutions to this particularly challenging problem.

In this thesis, we proposed two methods to generate skeletal structures of trees from TLS data. An especially decisive aspect during development has been the applicability of the methods to forest scenes with comparably high stand density in contrast to a multiscan that concentrates on a single tree.

Due to the lack of a comprehensive solution definition, we initially compiled a number of properties that the prospective methods and respectively their results should have in section 2.5. Comparing the created methods and the experiment results to the stated requirements, we conclude that both solutions confirm to them to a large extent.

The first method that is explained in chapter 5 generates a skeleton from a tree data set fully automatically, with few control parameters and in short time. A variation of the Circle Hough Transform [Duda-1972] is proposed and used in a scheme with RANSAC [Fischler-1981] to trace the trunk centerline. Subsequently, processing is performed on a voxel space representation of the data. A greedy search is started at each of the voxels that supposedly represent branch tips to find a path to the trunk through the voxel space. Evidently, it is a working solution to the problem. However, the particular result of the method strongly depends on the content of the 3D point cloud that is delivered as input. In other words, the scan coverage has a high impact on the overall result. In particular the voxel space structure that is imposed onto the data has ambivalent qualities: On the one hand, the adjacency relationships that are introduced make the retrieval of spatial structure possible at all. On the other hand, the same mechanism induces a potentially spurious topology.

At present, the evaluation of the retrieved skeletons is limited to visual assessment. We found that the representation is often not a description of the full tree that is discernible in the 3D point cloud. Branches are disconnected from the main tree component in voxel space. Therefore, they are missing in the skeleton. In some cases, the skeleton does not represent only a single tree. Due to the high stand density on the study site, tree crowns are intertwined, which is eventually reflected by the retrieved skeleton. Moreover, the skeleton vertices are centered in the point group that was contained within a particular voxel, but centering the skeleton globally in the point cloud is not enforced.

The method is a fast tool to retrieve the main geometric features and to generate a 3D line graph from the data set that approximates the spatial tree structure. Although the method functions adequately and consequently represents a solution to the given problem, we found that the approach is in general rather limited. This reasoning provoked the revision of the problem's starting point, which led to an approach with completely different characteristics than the first one.

The second method integrates a novel strategy to organize all attributes of TLS data from a single scan, which is discussed in chapter 6. The raster alignment opened up interesting possibilities to analyze the data and is the basis for the retrieval of skeletal structures. An efficient, yet rather simple strategy is proposed to trace boundaries including depth discontinuities of connected components that were detected with Connected Component Labeling [Shapiro-2001] in chapter 6. The obtained boundary information is then input to a routine that estimates the 2D skeleton of the connected component via the Voronoi Diagram [Aurenhammer-2000]. After refinement of the initial skeleton, the component is decomposed into parts that represent branches or the trunk. Each element of the component directly translates to a 3D point. The decomposition of the component shape therefore represents a segmentation of the corresponding 3D point set. The point distribution of each segmented subset is summarized as a Principal Curve in form of a polyline using the Polygonal Line Algorithm [Kégl-1999]. The result of the method is a set of polylines that reflects the distribution of the component's 3D point set with a comparably high degree of detail.

The presented method is fully automatic and exploits the 2D-3D coordinate relation of raster elements in a clever and efficient way in the course of processing. The key idea of Principal Curves is the summarizing of a point set's distribution as a space curve that traces along the "middle" of a point set [Kégl-2002]. Evidently, the polylines that are generated by the Polygonal Line Algorithm are therefore properly centered in the given point set. However, the point set needs to have such a "middle", otherwise the polyline is not the appropriate tool to simplify the point set to a line graph. The number of control parameters is higher in contrast to the first method, but the method consists also of more subprocedures and is more elaborate in general. The initially stated runtime limits are kept on average. In its current implementation, the runtime of the method exhibits a direct dependence on the size of the component that is processed. This is mainly due to the point location problem, which is solved in the method. But also the iterative nature of the Polygonal Line Algorithm leads to longer processing times if the point set is large. For this reason, the experiments had been limited regarding the allowed component size.

The obvious drawback of the second method is that partial skeletons are retrieved. Though, these partial skeletons represent in fact a large part of the branching structure in the tree crown in some cases. Generally speaking, the results of this method trace the perceptible shape of the 3D point sets with much more detail than the previously presented method. At the same time, the method is at present rather sensitive to noise effects. There are a number of issues that need to be implemented in a more sophisticated way as discussed in section 7.5. The method consists of an intricate combination of procedures, but it is founded on established concepts from image analysis like the Medial Axis. In contrast to the first method, the second method can be much better theoretically analyzed and consequently starting points for further developments are plentiful.

Ultimately, construction of full tree skeletons on this basis is imaginable: The partial skeletons that are of high reproduction accuracy could be considered as building bricks to a bottom-up assembling scheme to create a complete skeleton. We arrived at the conclusion that this is most likely a very promising way to achieve a robust solution to the problem of retrieving the spatial structure of trees from TLS data.

A POINT ATTACHMENT STRATEGY

In this section, we describe the strategy that is used to assign a skeleton node to each element of the connected component as explained in chapter 7. The basic task is summarized in algorithm 7.2 of section 7.2.2. In the experiments, a more sophisticated implementation has been applied to enhance performance. Another essential aspect was the clarity of the procedure: Since the actual implementation of algorithm 7.2 is cumbersome to debug in practice, we decided to decompose the general task into smaller problems that are in a sense simpler to solve one by one. First, we state the given problem. Second, we explain the solution that was implemented for the experiments. Last, we give a brief discussion of the applied strategy.

A.1 Problem Description

The given input is a graph G that is the union of the set of boundary nodes and the set of skeleton nodes. The set of boundary nodes represents the component's interior boundary and possible boundaries of holes that were determined as described in section 6.3.3. Each boundary vertex has 2D integer coordinates because it is also a raster element. Edges that belong to the main boundary form a closed curve. If the component has holes, boundary edges that form smaller closed curves are contained within the main boundary.

The set of skeleton nodes represents the 2D skeleton graph that was obtained in section 7.2.1. Each skeleton vertex has 2D coordinates in continuous Euclidean space. The 2D skeleton graph is not necessarily a tree. If the component has holes, they are most likely reflected by cycles in the skeleton graph. A skeleton edge can belong to the set of boundary elements at the same time as a result of the linking of sub graphs in section 7.2.1.

Beside the graph G, a set of points is given. Each point represents a raster element of the connected component. Therefore, each point has 2D integer coordinates and 3D coordinates in continuous Euclidean space that were measured during the scan. Consequently, raster elements that are part of the interior component boundary or a hole boundary are represented as boundary vertices and points at the same time.

The problem is that the points do not know about their location relative to any node of graph G. However, this information is essential to segment the point set on the basis of the skeleton graph. Hence, each point has to be assigned to its closest, appropriate skeleton node.

In order to solve this task, two aspects have to be taken into account: First, the point location problem needs to be treated. As stated in section 7.2.2, the shortest Euclidean distance between a point and a graph edge is the distance between the point and its projection point on the edge. If

the projection point is located outside the line segment that is represented by the graph edge, then the closest skeleton node to the point is one of the vertices of the edge or another edge respectively. Consequently, a point is assigned to a skeleton vertex only if it cannot be properly assigned to a skeleton edge.

Second, the assignment of a point to a skeleton node needs to be appropriate. Assigning the point to the closest skeleton node does not necessarily result in the segmentation that is desirable. Points may be located closer to a different branch in the skeleton graph than to the branch, to which they apparently belong (see figure 7.5). As a result, it has to be ensured that the connection line between a point and the projection point on the skeleton does not intersect the boundary. Exceptions are points that coincide with boundary vertices and points that are located on or inside boundary elements, where no skeleton part exists. The pruning of spurious edges from the skeleton graph (see section 7.2.1) removes skeleton edges that reach into component regions, which represent small twigs or perturbations of the boundary. These regions are often connected to the main body of the component by a narrow corridor. Straight projection lines that connect points located in these regions to skeleton nodes inevitably intersect the boundary, although the assignment to the particular node is correct.

A.2 Method

Our solution to the given problem is inspired by the partitioning strategy that was applied by [Kégl-2002]: First, we try to assign a point to a skeleton edge. Only if no suitable candidate in the edge set can be found, we test the set of skeleton vertices to find a suitable assignment for the point.

We decompose the given problem into subtasks that are solved one after another. The basic idea is to partition the component shape into polygons. The partitioning integrates skeleton nodes and boundary nodes. Afterwards, it the set of skeleton nodes that is nearest to a particular polygon is known. For each point, we query among the set of polygons to find the one, which contains the particular point. Finally, each polygon has a set of points and a set of close skeleton nodes. For each point in a polygon, we test the nodes of the corresponding set of skeleton nodes and assign the point to the node that minimizes the 2D Euclidean distance to this point. In the following, we explain the subtasks in detail.

Enforcement of subdivision constraint

As a first step, we ensure that each edge of the skeleton graph has only one adjacent vertex v of $deg(v) \neq 2$. If both vertices of an edge are not regular line vertices i.e. they are both of a degree different from 2, the edge is split midway and a new skeleton vertex is added accordingly.

Component partitioning

We consider only vertices of $deg(v) < 3$ in this subprocedure. A line vertex of $deg(v) = 2$ has two angles that are spanned by its two incident edges. Each angle is treated separately: We calculate the bisector vector that subdivides the angle in two equal parts. Starting at the vertex position, we trace along the direction of the bisector vector until an intersection with a node of graph G occurs. If the hit node is a boundary node, a new edge connecting the vertex to the point of intersection is added to the graph. The point of intersection is integrated as a new boundary vertex accordingly.

If the hit node is a skeleton node, the bisector is discarded. If no intersection occurred until a maximum distance t_d has been traced, the bisector is discarded as well.

A leaf, i.e. a vertex of $deg(v) = 1$ has only one incident edge. In this case, we trace along the direction of the normal vector of the incident edge in both directions in the same way.

After treating all considered vertices, the graph G consists of skeleton nodes, boundary nodes, and partition edges that may intersect each other. In order to restore the planarity of the graph, these intersections of partition edges are integrated into the graph. In other words, an intersection of two partition edges is properly replaced by four edges and a new partition vertex.

The graph G represents a partitioning of the component into polygons because each atomic cycle of the graph's rotation system is a polygon (see section 6.3.2). Consequently, we build a rotation system of graph G to retrieve the information about polygons and their adjacency relations among each other. The result is basically a list of polygons. Furthermore, the neighboring polygons to a particular polygon are known as well.

Propagation of skeleton node location among polygons

A number of polygons has an edge or vertex that is a skeleton node. The information about the relative location of skeleton nodes is propagated iteratively among the polygons.

Each polygon has a set N_{skel} of skeleton nodes that are in its proximity. First, each polygon is checked whether it has edges or vertices that are skeleton nodes. These nodes are added to the set N_{skel}. Polygons that are adjacent to skeleton nodes are of first class.

Second, each polygon is tested whether it has polygons of first rank in its adjacency. If yes, the current polygon's set N_{skel} obtains the union of all sets N_{skel} of neighboring polygons that are of first class. The polygon is then of second class.

Iteratively, each polygon is tested whether it has polygons in its adjacency that have been ranked in a previous iteration, which implies that they are located nearer to skeleton nodes than the currently considered polygon. The iteration is stopped when all polygons have been ranked or if no further change of classes occurred.

As explained in section 6.3.2, the boundary can form smaller polygons that have no adjacency beside a bridge that connects it to another boundary element. In this case, the set N_{skel} from the next reachable polygon is propagated.

Point location querying

Beside the set N_{skel} of nearest skeleton nodes, a polygon also has a set M_P of points. Each point is tested against the axis aligned bounding box of each polygon. If a point is located within a polygon's bounding box, a precise point-in-polygon test is performed. If the point is truly located within the considered polygon, the point is added to the polygon's point set M_P. Boundary vertices are points as well. They are directly assigned to the set M_P of the adjacent polygon.

Clearly, each point can only belong to one polygon. The result of the subprocedure is therefore a proper partition of the point set.

Assignment of points to skeleton nodes

Each polygon has a set of M_P of points and a set N_{skel} of nearby skeleton nodes. For each point $p \in M_P$, we find the skeleton node $s \in N_{skel}$ that minimizes the 2D Euclidean distance to the point. Finally, the point is assigned to the determined skeleton node.

A.3 Discussion

The presented strategy to implement the point attachment task results in a proper assignment of points to skeleton nodes in the majority of cases. However, not all situations are handled adequately. Most importantly, we currently neglect vertices v of $deg(v) > 2$, which can lead to an undesirable assignment for small subsets of points. At present, the integration of these vertices in the partition routine of section A.2 seems too expensive in comparison to the improvement that is gained by it. The constraint that is described in section A.2 was introduced instead to confine a potentially unsatisfying assignment to a small region of the component around a vertex v of $deg(v) > 2$.

At present, the task is solved in an adequate way. As pointed out previously, our focus was on replacing the monolithic formulation given in algorithm 7.2 by a solution that maintains clarity with regard to an actual implementation. Therefore, we decided to decompose the big problem into smaller subtasks following a divide-and-conquer philosophy.

Despite of being only a prototypical implementation, the point attachment procedure is rather quick on average as the experiments showed. However, we noticed that the performance degrades severely if the component size increases considerably. Due to this behavior we excluded components that exceed 100,000 elements in size. These components mostly represent large patches of trunk and only rarely branching structure. Evidently, the proposed strategy needs to be enhanced and optimized to find a solution with high performance for the given problem.

Clearly, this is not the only possible solution to the given problem. A less complex and more elegant algorithm to solve the task would be favorable.

B GLOSSARY & ACRONYMS

B.1 Glossary

Adjacency The adjacency of an object comprises neighboring objects that share certain structures. In case of a 3D voxel space, the 26-adjacency of a voxel consists of all neighboring voxels that share a border, corner or wall with the considered voxel. In a 2D grid, the 4-adjacency of a raster element comprises elements that share a border with the considered element; the 8-adjacency comprises elements that share a border or a corner with the considered element. In a graph, the adjacency of a vertex comprises the vertices that are endpoints of the edges that are incident to the considered vertex.

Adjacency graph From a given point set, an adjacency graph can be created by interpreting the points as vertices and inserting edges between them. Often, edges between a point and each one of the k nearest neighbors of the considered point are inserted. Alternatively, edges can be created between a point and each point that is located within a certain radius around the considered point. Instead of individual points, vertices may also represent point groups. The definition depends on the particular application context.

Centroid The geometric centroid of a set of points is the mean value of the coordinates of the points.

Connected component A connected component is a set of elements in a grid where each pair of elements is mutually reachable w.r.t. the applied definition of *connectedness*. In a binary image, a connected component is commonly a set of white pixels on black background pixels, where each pair of adjacent pixels is white. In a voxel space, a pair of voxels is reachable if a sequence of voxels between them can be found, where each two successive voxels are *connected* (see section 2.3).

Constrained confirming Delaunay triangulation A constrained confirming Delaunay triangulation is a Delaunay triangulation that maintains a predefined set of edges between vertices. In order to satisfy the Delaunay criterion for each of the triangles, a predefined edge may be split up into several smaller edges. The resulting newly inserted vertices can be used to create conforming triangles.

Convex hull The convex hull of a finite set of 2D points consists of the subset of points that form a convex polygon in which all remaining points of the set are located. The notion can be extended to 3D accordingly.

Degree The degree (or valence) of a vertex is the number of edges that are incident to the considered vertex, e.g. a leaf vertex v has $deg(v) = 1$.

Dendrometry Dendrometry is the measurement of geometrical properties of plants like for instance plant height or diameter of extremities.

Edge An edge is an atomic unit of a graph that connects exactly two vertices. If coordinates are associated with the vertices, the edge can be interpreted as a line segment and drawn onto a 2D plane or rendered in 3D space, respectively. An edge can be annotated by properties such as the length of the line segment or a set of points.

Graph A graph consists of a set of vertices and a set of edges that connect pairs of vertices.

K nearest neighbor strategy Since the a point set has no inherent adjacency relations, the adjacency of a point is defined as the k points that are nearest to the considered point if ordered by their Euclidean distance. The number of points K is a parameter that needs to be given.

Kmeans k-means clustering is a well-known clustering algorithm that partitions a given set of points into k disjoint subsets. Each point is assigned to the cluster with the nearest mean value, see e.g. [Weisstein-2014a].

Leaf In a graph, a vertex that has only a single incident edge is called a leaf.

Multiscan A multiscan is a short term to convey the notion of a scan project that consists of several co-registered scans of the same scene.

Octree An octree is a tree data structure, where each vertex in the tree has exactly eight children except the leaves. Different levels of detail can be represented within this data structure because the level of subdivision among the vertices can vary.

Phyto-element A phyto-element is any part of a tree like a branch, trunk or leaf.

Polyline A polygonal line (polyline for short) is a sequence of line segments.

RANSAC RANSAC (Random Sample Consensus) [Fischler-1981] is a robust estimation scheme to find a set of inliers from given data that complies with a given model.

Rotation system A rotation system is a cyclic permutation of edges at each vertex that can be used to traverse all possible cycles in a given graph [Gross-1987], which is explained in section 6.3.2.

Topology "Topology is the mathematical study of the properties that are preserved through deformations, twistings, and stretchings of objects. Tearing, however, is not allowed." [Weisstein-2014b] For instance, a tea cup with a handle and a torus are topologically equivalent objects. In other words, topology can be in investigated without considering the geometric coordinates of the elements. But geometry cannot be examined without the topology of the elements.

Tree graph A tree is a graph that contains no cycles.

Vertex A vertex is an atomic unit of a graph. In the thesis, we associate coordinates in 2D or 3D with a graph vertex. In addition, a vertex can be annotated with additional properties.

Voxel An element of the voxel space is called a voxel. Commonly, a voxel has cuboid shape and is annotated by attributes as for instance the number of 3D points that were mapped to it. A voxel with an empty point set is referred to as an empty voxel. A voxel has 3D integer coordinates that indicate its position within the voxel space.

Voxel space A voxel space is a partition of a region of continuous Euclidean 3D space into a regular spatial orthogonal array. Each array element is called a voxel. A 3D point set can be mapped to voxel space by a transformation as given in section 5.2.3.

B.2 Acronyms

2D	Two dimensional.
3D	Three dimensional.
ALS	Airborne laser scanning.
CCL	Connected Component Labeling.
CHT	Circle Hough Transform.
DAAD	Deutscher Akademischer Austauschdienst (German Academic Exchange Serive).
DBH	Diameter at breast height.
DFG	Deutsche Forschungsgemeinschaft (German Research Foundation).
DHT	Disc Hough Transform.
DTM	Digital terrain model.
I/O	Input/ output.
IAL	Indexed Attribute List.
Laser	Light amplification by stimulated emission of radiation.
LiDAR	Light detection and ranging.
MLS	Mobile laser scanning.
MSE	Mean square error.
MST	Minimal spanning tree.
OEM	Original equipment manufacturer.
PCS	Project coordinate system.
PLA	Polygonal line algorithm.
RGB	Color model with channels red, green, blue.
SCA	Space colonization algorithm.
SOCS	Scanner's own coordinate system.
TLS	Terrestrial laser scanning.
TOF	Time of flight.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

BIBLIOGRAPHY

[Aschoff-2004a] T. Aschoff and H. Spiecker, "Algorithms for the automatic detection of trees in laser scanner data", *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. 8/W2, pp. 71–75, 2004.

[Aschoff-2004b] T. Aschoff, M. Thies, and H. Spiecker, "Describing forest stands using terrestrial laser-scanning", *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 35, no. B5, pp. 237–241, 2004.

[Aurenhammer-2000] F. Aurenhammer and R. Klein, "Voronoi Diagrams", in *Handbook of Computational Geometry*, North-Holland Publishing, 2000, pp. 201–290.

[Azmy-2012] S. N. Azmy *et al.*, "Counting in the dark: non-intrusive laser scanning for population coutning and identifying roosting bats", *Scientific Reports*, vol. 2, no. 524, p. 4, 2012.

[Biasotti-2008] S. Biasotti *et al.*, "Skeletal Structures", in *Shape Analysis and Structuring*, Springer, 2008, pp. 145–183.

[Bienert-2010] A. Bienert *et al.*, "Voxel space analysis of terrestrial laser scans in forest for wind field modelling", *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 38, no. 5, pp. 92–97, 2010.

[Bienert-2013] A. Bienert and D. Schneider, "Range image segmentation for tree detection in forest scans", *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 2, no. 5/W3, pp. 49–54, 2013.

[Bienert-2014] A. Bienert, C. Hess, H.-G. Maas, and G. von Oheimb, "A voxel-based technique to estimate the volume of trees from terrestrial laser scanner data", *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, no. XL-5, pp. 101–106, 2014.

[Blaskow-2014] R. Blaskow and D. Schneider, "Analysis and correction of the dependency between laser scanner intensity values and range", *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, no. XL-5, pp. 107–112, 2014.

[Blum-1967] H. Blum, "A transformation for extracting new descriptor of shape", *Models for the perception of speech and visual form*, vol. 19, no. 5, pp. 362–380, 1967.

[Boehler-2003] W. Boehler, A. Marbs, and M. Bordas Vicent, "Investigating Laser Scanner Accuracy", *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 34, no. 5, pp. 696–701, 2003.

[Boost] *Boost c++ libraries*. [Online]. Available: `http://www.boost.org/` (visited on 08/28/2013).

[Bremer-2013] M. Bremer, M. Rutzinger, and V. Wichmann, "Derivation of tree skeletons and error assessment using LiDAR point cloud data of varying quality", *ISPRS Journal of Photogrammetry and Remote Sensing*, no. 80, pp. 39–50, 2013.

[Bucksch-2008] A. Bucksch and R. Lindenbergh, "Campino - A skeletonization method for point cloud processing", *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 63, no. 1, pp. 115–127, 2008.

[Bucksch-2010] A. Bucksch, R. Lindenbergh, and M. Menenti, "SkelTre - Robust skeleton extraction from imperfect point clouds", *Visual Computer*, vol. 26, no. 10, pp. 1283–1300, 2010.

[Bucksch-2011] A. Bucksch, "Revealing the skeleton from imperfect point clouds", Disseration, Delft University of Technology, Delft, The Netherlands, 2011.

[Cheng-2006] Z. Cheng, X. Zhang, and T. Fourcaud, "Tree Skeleton Extraction from a Single Range Image", in *Proceedings of the Second International Symposium on Plant Growth Modeling and Applications*, 2006, pp. 274–281.

[Cheng-2007] Z.-L. Cheng, X.-P. Zhang, and B.-Q. Chen, "Simple reconstruction of tree branches from a single range image", *Journal of computer science and technology*, vol. 22, no. 6, pp. 846–858, 2007.

[Chernov-09a] N. Chernov, *Circle fit (kasa method)*, 2009. [Online]. Available: `http://www.mathworks.com/matlabcentral/fileexchange/22642` (visited on 03/16/2014).

[Chernov-09b] ——, *Circle fit (pratt method)*, 2009. [Online]. Available: `http://www.mathworks.com/matlabcentral/fileexchange/22643` (visited on 03/16/2014).

[Chmielewski-2010] L. J. Chmielewski *et al.*, "Fuzzy Hough Transform-Based Methods for Extraction and Measurements of Single Trees in Large-Volume 3D Terrestrial LIDAR Data", in *Computer vision and graphics*, vol. 6374, Springer, 2010, pp. 265–274.

[Cornea-2007] N. D. Cornea, D. Silver, and P. Min, "Curve-Skeleton properties, applications and algorithms", *IEEE Transactions on Visualizations and Computer Graphics*, vol. 13, no. 3, pp. 530–548, 2007.

[Côté-2011] J.-F. Côté, R. A. Fournier, and R. Egli, "An architectural model of trees to estimate forest structural attributes using terrestrial LiDAR", *Environmental Modelling & Software*, vol. 26, no. 6, pp. 761–777, 2011.

[Dai-2010] M. Dai, H. Li, and X. Zhang, "Tree Modeling through Range Image Segmentation and 3D Shape Analysis", in *Advances in Neural Network Research & Applications*, Springer, 2010, pp. 413–422.

[Delagrange-2011] S. Delagrange and P. Rochon, "Reconstruction and analysis of a deciduous sapling using digital photographs or terrestrial-LiDAR technology", *Annals of Botany*, vol. 108, no. 6, pp. 991–1000, 2011.

[Dijkstra-1959] E. W. Dijkstra, "A note on two problems in connexion with graphs", *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[Duda-1972] R. O. Duda and P. E. Hart, "Use of the Hough Transformation to detect lines and curves in pictures", *Graphics and Image Processing*, vol. 15, no. 1, pp. 11–15, 1972.

[Durrieu-2008] S. Durrieu *et al.*, "Spatial quantification of vegetation density from terrestrial laser scanner data for characterization of 3D forest structure at plot level", in *Proceedings of SilviLaser*, 2008.

[Eigen] *Eigen*. [Online]. Available: `http://eigen.tuxfamily.org/index.php?title=Main_Page` (visited on 04/08/2014).

[Eu-1994] D. Eu and G. T. Toussaint, "On approximating polygonal curves in two and three dimensions", *CVGIP: Graphical Models and Image Processing*, vol. 56, no. 3, pp. 231–246, 1994.

[Eysn-2013] L. Eysn *et al.*, "A practical approach for extracting tree models in forest environments based on equirectangular projections of terrestrial laser scans", *Remote Sensing*, vol. 5, no. 11, pp. 5424–5448, 2013.

[FAO-2010] FAO, *Key findings - global forest resources assessment*, 2010. [Online]. Available: `http://foris.fao.org/static/data/fra2010/KeyFindings-en.pdf` (visited on 03/12/2014).

[FAO-2011] ——, *State of the world's forests 2011.* FAO, 2011. [Online]. Available: `http://www.fao.org/docrep/013/i2000e/i2000e.pdf` (visited on 03/12/2014).

[Faro-2013] FARO Technologies, *Faro Laser Scanner Focus 3D: Features, Benefits & Technical Specifications*, 2013. [Online]. Available: `http://www2.faro.com/site/resources/share/944` (visited on 04/08/2014).

[Fischler-1981] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[Fleck-2007] S. Fleck *et al.*, "Terrestrial lidar measurements for analysing canopy structure in an old-growth forest", *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. 3/W52, pp. 125–129, 2007.

[García-2011] M. García *et al.*, "Terrestrial laser scanning to estimate plot-level forest canopy fuel properties", *International Journal of Applied Earth Obersvation and Geoinformation*, vol. 13, no. 4, pp. 636–645, 2011.

[Gatziolis-2010] D. Gatziolis, S. Popescu, R. Sheridan, and N.-W. Ku, "Evaluation of terrestrial LiDAR technology for the development of local tree volume equations", in *Proceedings of SilviLaser*, 2010, pp. 197–205.

[Giannitrapani-1999] R. Giannitrapani and V. Murino, "Three-Dimensional Skeleton extraction by point set contraction", in *Proceedings of the International Conference on Image Processing*, 1999, pp. 565–569.

[Gorte-2004a] B. Gorte and N. Pfeifer, "Structuring Laser-scanned trees using 3d mathematical morphology", *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 35, no. B5, pp. 929–933, 2004.

[Gorte-2004b] B. Gorte and D. Winterhalder, "Reconstruction of laser-scanned trees using filter operations in the 3D raster domain", *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. 8/W2, pp. 39–44, 2004.

[Gorte-2006] B. Gorte, "Skeletonization of Laser-Scanned Trees in the 3D Raster Domain", in *Innovations in 3D Geo Information Systems*, Springer, 2006, pp. 371–380.

[Gross-1987] J. L. Gross and T. W. Tucker, *Topological graph theory*. Wiley-Interscience, 1987.

[Hartley-2003] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2. Cambridge University Press, 2003.

[Hastie-1984] T. Hastie, "Principal Curves and Surfaces", PhD Thesis, Stanford University, California, USA, 1984.

[Hastie-1989] T. Hastie and W. Stuetzle, "Principal Curves", *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 502–516, 1989.

[Henning-2006] J. G. Henning and P. J. Radtke, "Ground-based laser imaging for assessing three-dimensional forest canopy structure", *Photogrammetric Engineering and Remote Sensing*, vol. 72, no. 12, pp. 1349–1358, 2006.

[Hilker-2010] T. Hilker *et al.*, "Comparing canopy metrics derived from terrestrial and airborne laser scanning in a douglas-fir dominated forest stand", *Journal of Trees*, vol. 24, no. 5, pp. 819–832, 2010.

[Hopkinson-2004] C. Hopkinson, L. Chasmer, C. Young-Pow, and P. Treitz, "Assessing forest metrics with a ground-based scanning lidar", *Canadian Journal for Forest Research*, vol. 34, no. 3, pp. 573–583, 2004.

[Hosoi-2013] F. Hosoi, Y. Nakai, and K. Omasa, "3-D voxel-based solid modeling of a broad-leaved tree for accurate volume estimation using portable scanning lidar", *ISPRS Journal of Photogrammetry and Remote Sensing*, no. 82, pp. 41–48, 2013.

[Hough-1962] P. V. C. Hough, "Method and means for recognizing complex patterns", 3069654A, US Patent 3,069,654, 1962.

[Huber-2011] D. Huber, "The ASTM E57 File Format for 3D Imaging Data Exchange", in *Proceedings of SPIE7864 on Three-Dimensional Imaging, Interaction, and Measurement*, 2011.

[IEEE754-2008] *754-2008 - IEEE Standard for Floating-Point Arithmetic*, 2008. [Online]. Available: `http://ieeexplore.ieee.org/servlet/opac?punumber=4610933` (visited on 02/28/2013).

[Isenburg-2013] M. Isenburg, "LASzip: lossless compression of LiDAR data", *Photogrammetric Engineering and Remote Sensing*, vol. 79, no. 2, pp. 209–2017, 2013.

[Jolliffe-2002] I. Jolliffe, *Principal Component Analysis*, 2. Springer, 2002.

[Kasa-1976] I. Kasa, "A curve fitting procedure and its error analysis", *IEEE Transactions on Instrumentation and Measurement*, vol. 25, no. 1, pp. 8–14, 1976.

[Kégl-1999] B. Kégl, "Principal curves: learning, design, and applications", PhD Thesis, Concordia University, Montreal, Quebec, Canada, 1999.

[Kégl-2002] B. Kégl and A. Krzyzak, "Piecewise linear skeletonization using principal curves", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 59–74, 2002.

[Király-2007] G. Király and G. Brolly, "Tree height estimation methods for terrestrial laser scanning in a forest reserve", *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. 3/W52, pp. 211–215, 2007.

[Klette-2004] R. Klette and A. Rosenfeld, *Digital geometry: geometric methods for digital picture analysis*. Morgan Kaufman Publishers, 2004.

[Kovesi-00] P. Kovesi, *MATLAB and octave functions for computer vision and image processing*. [Online]. Available: `http://www.csse.uwa.edu.au/~pk/research/matlabfns/` (visited on 04/13/2011).

[LAS-2011] *LAS Specification Version 1.4 R12*, 2011. [Online]. Available: `http://www.asprs.org/a/society/committees/standards/LAS%5C_1%5C_4%5C_r12.pdf` (visited on 06/10/2012).

[Lefsky-2008] M. Lefsky and M. McHale, "Volume estimates of trees with complex architecture from terrestrial laser scanning", *Journal of Applied Remote Sensing*, vol. 2, no. 1, p. 19, 2008.

[Leica-2008] *Cyclone PTG File Format Specification - Version 1*, 2008. [Online]. Available: `http://www.xdesy.de/freeware/PTG-DLL/PTG-1.0.pdf` (visited on 02/27/2013).

[Leica-2012] Leica Geosystems, *Cyclone pointcloud export format - Description of ASCII .ptx format*, Knowledge Base Article, 2012. [Online]. Available: `http://www.leica-geosystems.com/kb/?guid=5532D590-114C-43CD-A55F-FE79E5937CB2` (visited on 01/09/2014).

[Li-2010] H. Li, X. Zhang, M. Jaeger, and T. Constant, "Segmentation of forest terrain laser scan data", in *Proceedings of the 9th ACM SIGGRAPH Conference on Virtual-Reality Continuum and its Applications in Industry*, 2010, pp. 47–54.

[Liang-2012] X. Liang *et al.*, "Automatic Stem Mapping Using Single-Scan Terrestrial Laser Scanning", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 2, pp. 661–670, 2012.

[Livny-2010] Y. Livny *et al.*, "Automatic Reconstruction of Tree Skeletal Structures from Point Clouds", *ACM Transactions on Graphics (TOG)*, vol. 29, no. 6, article no. 151, 2010.

[Maas-2008] H.-G. Maas, A. Bienert, S. Scheller, and E. Keane, "Automatic forest inventory parameter determination from terrestrial laserscanner data", *International Journal of Remote Sensing*, vol. 29, no. 5, pp. 1579–1593, 2008.

[Mäntylä-1988] M. Mäntylä, "Half-Edge Data Structure", in *An introduction to solid modeling*, Computer Science Press, Inc., 1988.

[Michel-2008] P. Michel *et al.*, "Assessing the ecological application of lasergrammetric techniques to measure fine-scale vegetation structure", *Ecological Informatics*, vol. 3, no. 4-5, pp. 309–320, 2008.

[Mohar-2001] B. Mohar and C. Thomassen, *Graphs on surfaces*. Johns Hopkins University Press, 2001.

[Moskal-2012] L. M. Moskal and G. Zheng, "Retrieving forest inventory variables with terrestrial laser scanning (TLS) in urban heterogeneous forest", *Remote Sensing*, vol. 4, no. 1, pp. 1–20, 2012.

[MVG-2004] A. Zisserman, *MATLAB Functions for Multiple View Geometry*. [Online]. Available: `http://www.robots.ox.ac.uk/~vgg/hzbook/code/` (visited on 11/29/2013).

[Ogniewicz-1995] R. L. Ogniewicz and O. Kübler, "Hierarchic Voronoi Skeletons", *Pattern Recognition*, vol. 28, no. 3, pp. 343–359, 1995.

[Petrie-2009a] G. Petrie and C. K. Toth, "Introduction to laser ranging, profiling, and scanning", in *Topographic Laser Ranging and Scanning: Principles and Processing*, J. Shan and C. K. Toth, Eds., CRC Press, 2009.

[Petrie-2009b] ——, "Terrestrial Laser Scanning", in *Topographic Laser Ranging and Scanning: Principles and Processing*, J. Shan and C. K. Toth, Eds., CRC Press, 2009.

[Pfeifer-2004a] N. Pfeifer, B. Gorte, and D. Winterhalder, "Automatic Reconstruction of single trees from terrestrial laser scanner data", *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 35, no. B5, pp. 114–119, 2004.

[Pfeifer-2004b] N. Pfeifer and D. Winterhalder, "Modelling of tree cross sections from terrestrial laser scanning data with free-form curves", *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. 8/W2, pp. 76–81, 2004.

[PLA-1999] B. Kégl, *The Java Implementation of the Polygonal Line Algorithm*. [Online]. Available: `http://www.iro.umontreal.ca/~kegl/research/pcurves/implementations/` (visited on 08/28/2013).

[Pratt-1987] V. Pratt, "Direct Least-Squares Fitting of Algebraic Surfaces", *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 145–152, 1987.

[Pretzsch-2011] H. Pretzsch, S. Seifert, and P. Huang, "Beitrag des terrestrischen laserscannings zur erfassung der struktur von baumkronen", *Schweizerische Zeitung für Forstwesen*, vol. 162, no. 6, pp. 186–194, 2011.

[Preuksakarn-2010] C. Preuksakarn *et al.*, "Reconstructing Plant Architecture from 3D Laser scanner data", in *Proceedings of the 6th International Workshop on Functional-Structural Plant Models*, 2010.

[Propastin-2013] P. Propastin and O. Panferov, "Retrieval of remotely sensed LAI using landsat ETM+ data and ground measurements of solar radiation and vegetation structure: implication of leaf inclination angle", *International Journal of Applied Earth Observation and Geoinformation*, vol. 25, pp. 38–46, 2013.

[Raumonen-2013] P. Raumonen *et al.*, "Fast Automatic Precision Tree Models from Terrestrial Laser Scanner Data", *Remote Sensing*, vol. 5, no. 2, pp. 491–520, 2013.

[Reulke-2005] R. Reulke and N. Haala, "Tree species recognition with fuzzy texture parameters", in *Combinatorial image Analysis*, Springer, 2005, pp. 607–620.

[Riegl-2014] Riegl, *Datasheet VZ-6000*. [Online]. Available: `http://www.riegl.com/uploads/tx_pxpriegldownloads/Datasheet_VZ-6000_2014-09-19.pdf` (visited on 11/04/2014).

[Runions-2007] A. Runions, B. Lane, and P. Prusinkiewicz, "Modeling Trees with a Space Colonization Algorithm", in *Proceedings of the Third Eurographics conference on Natural Phenomena*, 2007, pp. 63–70.

[Russell-2010] S. Russell and P. Norvig, "Searching", in *Artifical Intelligence*, 3, Prentice Hall, 2010, pp. 87–89.

[Sansoni-2009] G. Sansoni, M. Trebeschi, and F. Docchio, "State-of-the-art and applications of 3D imaging sensors in industry, cultural heritage, medicine, and criminal investigation", *Sensors*, vol. 9, no. 1, pp. 568–601, 2009.

[Schilling-2011a] A. Schilling, A. Schmidt, and H.-G. Maas, "Automatic Tree Detection and Diameter Estimation in Terrestrial Laser Scanner Point Clouds", in *Proceedings of the 16th Computer Vision Winter Workshop*, 2011, pp. 75–83.

[Schilling-2011b] A. Schilling, A. Schmidt, H.-G. Maas, and S. Wagner, "Topology Extraction using Depth First Search on Voxel Representations of Tree Point Clouds", *International Archives of Photogrammetry and Remote Sensing*, vol. 38, no. 5/W12, 2011.

[Schilling-2012a] A. Schilling, A. Schmidt, and H.-G. Maas, "Principal Curves for Tree Topology Retrieval from TLS Data", in *Proceedings of SilviLaser*, 2012.

[Schilling-2012b] ——, "Tree Topology Representation from TLS Point Clouds Using Depth-First Search in Voxel Space", *Photogrammetric Engineering and Remote Sensing*, vol. 78, no. 4, pp. 383–392, 2012.

[Schmidt-2012] A. Schmidt, A. Schilling, and H.-G. Maas, "A method for the registration of hemispherical photographs and TLS intensity images", *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 39, no. B5, pp. 245–249, 2012.

[Shan-2009] J. Shan and C. K. Toth, Eds., *Topographic Laser Ranging and Scanning: Principles and Processing*. CRC Press, 2009.

[Shapiro-2001] L. G. Shapiro and G. C. Stockmann, "Binary Image Analysis", in *Computer Vision*, Prentice Hall, 2001, pp. 69–75.

[Shewchuk-1996] J. R. Shewchuk, "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator", in *Applied Computational Geometry Towards Geometric Engineering*, 1996, pp. 203–222. [Online]. Available: `http://www.cs.cmu.edu/~quake/triangle.html` (visited on 09/10/2013).

[Siddiqi-2008] K. Siddiqi and S. M. Pizer, *Medial Representations Mathematics, Algorithms and Applications*. Springer, 2008.

[Simonse-2003] M. Simonse, T. Aschoff, H. Spiecker, and M. Thies, "Automatic determination of forest inventory parameters using terrestrial laserscanning", in *Proceedings of the ScandLaser Scientific Workshop on Airborne Laser Scanning of Forest*, 2003.

[Sonka-1998] M. Sonka, V. Hlavac, and R. Boyle, "Hough Transform", in *Image processing, analysis, and machine vision*, 2, PWS Publishing, 1998, pp. 163–173.

[Strahler-2008] A. H. Strahler *et al.*, "Retrieval of forest structural parameters using a ground-based lidar instrument (echidna)", *Canadian Journal of Remote Sensing*, vol. 34, no. S2, S426–S440, 2008.

[Su-2011] Z. Su *et al.*, "Skeleton extraction for tree models", *Journal of Mathematical and Computer Modelling*, vol. 54, no. 4-5, pp. 1115–1120, 2011.

[Thies-2004] M. Thies, N. Pfeifer, D. Winterhalder, and B. G. H. Gorte, "Three-dimensional reconstruction of stems for assessment of taper, sweep and lean based on laser scanning of standing trees", *Scandinavian Journal of Forest Research*, vol. 19, no. 6, pp. 571–581, 2004.

[Tomlow-2002] J. Tomlow, "La evolución de la innovación estructural de Gaudí. Los proyectos de la sede de la Misión Franciscana, la iglesia de la Colonia Güell y el templo de la Sagrada Familia.", *OP Ingenería y territorio*, vol. 59, pp. 48–58, 2002.

[Trimble-2013] Trimble, *Trimble Scene 5.2 Manual*, 2013.

[Van Laar-2007] A. Van Laar and A. Akça, *Forest mensuration*. Springer, 2007.

[Verroust-2000] A. Verroust and F. Lazarus, "Extracting skeletal curves from 3D scattered data", *The Visual Computer*, vol. 16, no. 1, pp. 15–25, 2000.

[vLeeuwen-2010] M. van Leeuwen and M. Nieuwenhuis, "Retrieval of forest structural parameters using LiDAR remote sensing", *European Journal of Forst Reseach*, vol. 129, pp. 749–770, 4 2010.

[Vonderach-2012] C. Vonderach, T. Vögtle, P. Adler, and S. Norra, "Terrestrial laser scanning for estimating urban tree volume and carbon content", *International Journal of Remote Sensing*, vol. 33, no. 21, pp. 6652–6667, 2012.

[Vosselman-2010] G. Vosselman and H.-G. Maas, Eds., *Airborne and Terrestrial Laser Scanning*. Whittles Publishing, 2010.

[Wagemans-2012] J. Wagemans *et al.*, "A Century of Gestalt Psychology in Visual Perception: I. Perceptual Grouping and Figure-Ground Organization", *Psychological Bulletin*, vol. 138, no. 6, pp. 1172–1217, 2012.

[Wang-2012] Y. Wang *et al.*, "Tree branching reconstruction from unilateral point clouds", in *Transactions on Edutainment VIII*, Springer, 2012, pp. 250–263.

[Watt-2005] P. J. Watt and D. N. M. Donoghue, "Measuring forest structure with terrestrial laser scanning", *International Journal of Remote Sensing*, vol. 26, no. 7, pp. 1437–1446, 2005.

[Weisstein-2014a] E. Weisstein, *K-means clustering algorithm*. [Online]. Available: `http://mathworld.wolfram.com/K-MeansClusteringAlgorithm.html` (visited on 05/09/2014).

[Weisstein-2014b] ——, *Topology*. [Online]. Available: `http://mathworld.wolfram.com/Topology.html` (visited on 05/09/2014).

[West-2009] P. W. West, *Tree and forest measurement*, 2nd ed. Springer, 2009.

[Williams-2005] A. Williams, S. Barrus, R. K. Morley, and P. Shirley, "An efficient and robust ray-box intersection algorithm", in *ACM SIGGRAPH 2005 Courses*, 2005.

[Xu-2007] H. Xu, N. Gossett, and B. Chen, "Knowledge and heuristic-Based Modeling of Laser-Scanned Trees", *ACM Transactions on Graphics*, vol. 26, no. 4, 2007.

[Yan-2009] D.-M. Yan *et al.*, "Efficient and robust reconstruction of botanical structure from laser scanned data points", in *Proceedings of the 11th IEEE International conference on Computer-Aided Design and Computer Graphics*, 2009, pp. 572–576.

[ZF-2005] Zoller+Fröhlich, *Z+F Imager 5006 Benutzerhandbuch Version 3.0*. 2008.

[ZF-2012] ——, *Z+F LaserControl*, 2012. [Online]. Available: `http://www.zf-laser.com/fileadmin/editor/Broschueren/Z_F_LaserControl_kompr.pdf` (visited on 04/08/2014).

[Zhu-2008] C. Zhu, X. Zhang, B. Hu, and M. Jaeger, "Reconstruction of Tree Crown Shape from Scanned Data", in *Technologies for E-Learning and Digital Entertainment*, 2008, pp. 745–756.