TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Markku Åkerblom

Quantitative Tree Reconstruction from Terrestrial Laser Scanning Data and Applications

Julkaisu 1547 • Publication 1547

Tampere 2018

Tampereen teknillinen yliopisto. Julkaisu 1547
Tampere University of Technology. Publication 1547

Markku Åkerblom

Quantitative Tree Reconstruction from Terrestrial Laser Scanning Data and Applications

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Sähkötalo Building, Auditorium SA203, at Tampere University of Technology, on the 25th of May 2018, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2018

Doctoral candidate:
Markku Åkerblom
Laboratory of Mathematics
Faculty of Natural Sciences
Tampere University of Technology
Finland

Supervisor:
Mikko Kaasalainen, professor
Laboratory of Mathematics
Faculty of Natural Sciences
Tampere University of Technology
Finland

Instructor:
Pasi Raumonen, senior research fellow
Laboratory of Mathematics
Faculty of Natural Sciences
Tampere University of Technology
Finland

Pre-examiners:
Martin Herold, professor
Wageningen University & Research
Netherlands

Heikki Haario, professor
School of Engineering Science
Lappeenranta University of Technology
Finland

Opponent:
Marko Vauhkonen, professor
Department of Applied Physics
University of Eastern Finland
Finland

ISBN 978-952-15-4140-7 (printed)
ISBN 978-952-15-4147-6 (PDF)
ISSN 1459-2045

Abstract

Understanding the structure and dynamics of trees and forest is key in studying the environment and understanding current and future climates. Development has been fast in measurement technology for these purposes, as it is currently possible to measure forest terrestrially with photography-based instruments or either static or mobile laser scanning, and airborne using drones, helicopters or aeroplanes, and even from space using satellite-mounted instruments. However, as all these measurements are indirect presentations of the key attributes to study, they require powerful analysis methods to accompany them. This thesis focuses on terrestrial laser scanning data and presents a method for reconstructing comprehensive, quantitative structure models of trees from such data. The method is designed to be a tool for understanding tree and forest structure, as well as, dynamics and functionality, without the need for destructive measurements. The reconstructed models provide access to tree attributes previously impossible or laborious to measure, either at a single tree-scale, at forest-plot-scale or even at forest-scale. The thesis will present the reconstruction method and will focus on two of its applications: automatic tree species recognition and augmenting the produced structure models with leaves or needles, enabling more accurate simulations involving light propagation and plant interaction with the atmosphere. Additionally, parts of the thesis describe forms of dissemination used to promote the reconstruction method and its applications, increasing the rate of adoption into operational use. The dissemination approaches include several animations, interactive 3D models and open-source software.

Preface

I remember starting working on the tree reconstruction project, when the first version of the tree reconstruction procedure had been written. It was in the very first workshop I attended as part of this project, and after the presentation by my colleague, an emeritus professor told us quite bluntly that, he had tried to reconstruct trees from terrestrial laser scanning data, and thus knew for a fact, that it simply could not be done! And this was despite the results we were already presenting. For many years after that, I still had that doubt in the back of my head — at least every once in a while. On the other hand, I was constantly told by my supervisor Mikko Kaasalainen that we are going to change the information paradigm, and that accurate tree models — which we started calling QSMs — are the key in this revolution. Only as late as 2016, when our research group organized a workshop on tree data and modelling, did I really understand the effect we had made; A room-full of researchers from all-over the world, using the term QSM as it was something that had been taught in school for decades, and even more so, their believe that QSMs are the future, coming up with dozens of ideas on how they will be utilized. So the revolution is actually happening as predicted and I'm proud having played at least a small part in it.

> “ When life gives you lemonade, you make it into lemons — and life's going to be all like *whaaaaat?*
>
> *Phil Dunphy*

I started the dissertation project without a lot of knowledge about the world of academia, and it seems that with the project now completing, I'm left with even more questions. It might be an overall feature of the era we are in, but it seems to me that there is constant change going on at least in the Finnish academic institutions and atmosphere. Although it is said that evolution is key in success, I do feel that some level of permanence is also required to know if any change was for the better and where to move next. This is in fact the basic principle of mathematical optimization. Whatever the future might hold for scientific research in Finland, I do hope that the appreciation of science itself does not deteriorate, and that the appeal of this line-of-work does build up, because especially in the post-factual world we need facts and people who separate — and teach to separate — fact from fiction.

During my PhD studies I've had the pleasure of meeting a lot of people from different sectors of academia and various industries, many of whom have been, and continue to be, very passionate about the aspect of the world of science closest to them, but underestimating the importance of all other aspects. While I admire the intensity of their commitment, I do worry that it is just one more symptom of the growing binary view of everything. Inferring that one has to be better, pure mathematics or applied, or that either teaching or research is more important, mathematics is either embedded in everything or not useful for anything, research versus dissemination, theoretical versus computational, industry collaboration or independent science. It makes me think of a quote about only mad men dealing in absolutes but contrary to my recollection, such quote only exists in the realm of science fiction. Even so, I wish for a time in the future where we can again understand the need for compromise and not treat it as a curse word — because what is compromise other than a synonym for empathy?

> “There is no absolute point of view from which real and ideal can be finally separated and labelled.
>
> *T. S. Elliot*

Quite a large portion of my post-graduate time was spent on dissemination related aspects of research, and often the assumption seems to be that when you do something like that, you are choosing appearance over substance. However, I see research as a two part job, where you first make a discovery and then report it. Do either one alone and you are not a researcher, so why not give attention to both parts? Many people, myself included, have questioned the relevance and importance of dissemination-focused efforts during my studies, but I’ve come to the conclusion that in the end, the efforts were certainly worth it. An animation or an interactive user-interface demonstration is not meant to replace written publications, but they can be used to augment and elevate them, and to help new readers to find written publications. Furthermore, it is about acknowledging that people learn in different ways: some by reading text or equations, other by listening to oral presentations, and others by trying out things by themselves. Therefore, why not help as many people understand your work as possible? So I would like to challenge everyone — next time you make a discovery — take some time to think about how to report it in innovative ways, to ensure it gets the attention it deserves. But it should still be kept in mind that in a car race it is better to bet on the car with the best engine, not the one with the shiniest gloss.

I am thankful for my supervisor Mikko Kaasalainen for the words of encouragement, guidance and level of freedom I was given, and to my instructor Pasi Raumonen for the support, collaboration and discussions. The funding provided by the Academy of Finland through the Centre of Excellence in Inverse Problems Research and later the Centre of Excellence in Inverse Modelling and Imaging, allowed me to focus solely on research. I’m also thankful for the additional funding provided by Jenny and Antti Wihuri Foundation, Emil Aaltonen Foundation and Finnish Foundation for Technology Promotion.

I have been pleasantly surprised and grateful for the positive reception and inclusion by the forest research community over the years. In particular I would like to thank Eric Casella, Sanna Kaasalainen, Jari Liski, Raisa Mäkipää, and Risto Sievänen for their input and support.

I thank the pre-examiners of this thesis, professors Heikki Haario and Martin Herold, for their time and contributions.

To my *workplace proximity associates* Elina Viro, Henri Riihimäki, Kalle Rutanen, Juho Lauri, Hari Nortunen and Petteri Laakkonen, together with the first people at the department who where always nice to me, Tiina Sävilahti and Riitta Lahti, I thank you for everything we have shared and endured together.

Thank you to my friends Matti Javanainen, Suvi Lehtimäki, Jouni Mäkitalo, Tomi Turtiainen, Topi Uusitalo, Aino and Toni Vettenranta, and Jesse Vilja for all the work counterbalance activities and to some of you for efforts to — involuntarily on my part — conquer the beer capitols of the world. And to Kirsi, Matti, Laura and Kielo Virkki, thank you for letting me be a part of your life and for giving me one of the greatest honours of my life.

Carlos, my brother, thank you for your friendship, support and what must have been a billion jokes we've had the pleasure of sharing. Thank you for — quite literally — always being there for me. Also, I see your master's degree and raise you one doctoral degree.

> “Each friend represents a world in us, a world not born until they arrive, and it is only by this meeting that a new world is born.
>
> *Anaïs Nin*

Koen olevani vähintään keskiverto kirjoitetun sanan osa-alueella. Tiedän, että silti en pysty tuottamaan sanoja, joilla saisin teidät äiti, isä, Riikka ja Juhani vakuutettua siitä, että saavutukseni ovat teidän kovan työnne, esimerkkinne ja loputtoman tukenne ansiota. Teiltä olen vuosien saatossa oppinut enemmän työetiikasta, oikeudenmukaisuudesta ja ennen kaikkea periksiantamattomuudesta kuin kaikkien opintojeni aikana yhteensä. Kiitos että olette olleet jättiläisiä, joiden harteilta tämän kääpiön on ollut helppo ponnistaa!

My wife Fanni, as you are the brightest light in my life, they should name a star after you. However I am just a mathematician and not in the business of naming stars, and thus I had make stuff up to name in you honour. Thank you for being an inspiration and a role model and for standing by me for our first ten amazing years, through the good times and the tough times, and especially through efforts required to make this thesis into reality. You make me a better person, just as you seem to do everyone around you. ♡

Tampere

May, 2018

Markku Åkerblom

Contents

NOMENCLATURE

Notation	Description
a	Scalar number a.
$\mathbf{a}$	A vector.
M	A matrix.
$\mathbb{N}$	Set of natural numbers.
$\mathbb{Z}$	Set of integers.
$\mathbb{R}$	Set of real numbers.
$\{a_1, a_2, \dots\}$	A set of elements.
$[a_1, a_2, \dots]$	A list of elements.
$a \in A$	a is an element of set A.
$\|\| \mathbf{a} \|\|$	Euclidean norm of vector $\mathbf{a}$.
$\|a\|$	Absolute value of the scalar a.
$\forall$	Universal quantification.
$\exists$	Existential quantification.
$f(x)$	f is a function of x.
$=$	Equal to.
$\neq$	Not equal to.
$<$	Less than.
$\leq$	Less or equal than.
$>$	Greater than.
$\geq$	Greater or equal than.
$\approx$	Approximately equal to.
$\propto$	Proportional to.
$\pm$	Plus or minus depending on a condition.

Abbreviations

AGB	above-ground biomass
API	application programming interface
AR	augmented reality
CCC	concordance correlation coefficient
CV(RMSE)	coefficient of variation of the root-mean-square error
DBH	diameter at breast-height
DOI	digital object identifier
FaNNI	Foliage and Needles Naïve Insertion
GPR	ground-penetrating radar
HSL	hyper-spectral lidar
HTML	hypertext markup language
JSON	JavaScript object notation
LADD	leaf area density distribution
LAI	leaf area index
LS	least-squares
LiDAR	light detection and ranging
PCA	principal component analysis
PHP	PHP: hypertext preprocessor
QR	quick response
QSM	quantitative structure model
RANSAC	random sample consensus
RMSE	root-mean-square error
SQL	structured query language
SSM	stochastic structure model

SVM	support vector machine
TLS	terrestrial laser scanning
UI	user interface
VR	virtual reality
WebGL	web graphics library

List of publications

This dissertation is a compilation of five (5) peer-reviewed publications, of which four (4) are journal articles, one (1) article appears in a conference proceedings. The publications are listed below in chronological order. Description of the division of responsibilities is reported separately for each publication. The quick response (QR) codes contain the digital object identifier (DOI) web address of the respective publication.

Article I Åkerblom, M., Raumonen, P., Kaasalainen, M., Kaasalainen, S., and Kaartinen, H. 2012. Comprehensive quantitative tree models from TLS data. *IEEE International Geoscience and Remote Sensing Symposium Proceedings*.

M. Åkerblom developed the validation approach and the gap filling procedure for the reconstruction method, computed the validation results and drafted main parts of the manuscript. P. Raumonen reconstructed the example tree model, for which S. Kaasalainen and H. Kaartinen provided the data. All co-authors helped draft the manuscript.

Article II Raumonen, P., Kaasalainen, M., Åkerblom, M., Kaasalainen, S., Kaartinen, H., Vastaranta, M., Holopainen, M., Disney, M., and Lewis, P. 2013. Fast automatic precision tree models from terrestrial laser scanner data. *Remote Sensing*, 5(2):491–520.

P. Raumonen developed the TreeQSM reconstruction method and drafted main parts of the manuscript. M. Åkerblom developed the gap filling procedure and helped otherwise the development of the reconstruction method. S. Kaasalainen and H. Kaartinen provided TLS data, M. Vastaranta and M. Holopainen performed the validation tree branch measurements, M. Disney and L. Lewis provided the virtual tree validation data. All co-authors helped draft the manuscript.

Article III Åkerblom, M., Raumonen, P., Kaasalainen, M., and Casella, E. 2015. Analysis of geometric primitives in quantitative structure models of tree stems. *Remote Sensing*, 7(4):4581–4603.

M. Åkerblom created the stem models for the generated data section, reconstructed the models for both data types, computed the results, created the figures and wrote the main text. P. Raumonen created the reconstruction procedure, guided the ideas of this manuscript, wrote parts of the text and gave feedback on it. M. Kaasalainen supervised the whole process from the idea-stage to the final manuscript, wrote parts of the text and gave vital feedback on rest of the text. E. Casella carried-out the in-situ measurements, wrote the description of the measurements, and gave feedback on the manuscript.

Article IV Åkerblom, M., Raumonen, P., Mäkipää, R., and Kaasalainen, M. 2017. Automatic tree species recognition with quantitative structure models. *Remote Sensing of Environment*, 191:1 – 12.

M. Åkerblom selected the included classification features, carried-out the computations, drafted main parts of the manuscript, and produced the supplementary animations. R. Mäkipää provided the point cloud data and wrote descriptions of the forest plots. P. Raumonen reconstructed the tree models. All co-authors gave feedback on the manuscript.

Article V Åkerblom, M., Raumonen, P., Casella, E., Disney, M., Danson, F.M., Gaulton, R., Schofield, L.A., and Kaasalainen, M. 2018. Non-Intersecting leaf insertion algorithm for tree structure models. *Interface Focus*, 8(2).

M. Åkerblom developed the FaNNI algorithm, wrote the implementation, carried-out the computations, and drafted the manuscript. E. Casella acquired the TLS measurements, computed the QSMs and led the destructive leaf sampling experiment. M. Disney, F.M. Danson, R. Gaulton, and L.A. Schofield participated in that experiment. All co-authors helped draft the manuscript.

Supporting material

A key part of the thesis research has been the use of innovative dissemination methods, in an effort to increase the societal and scientific impact of the research. Dissemination of scientific results has become increasingly important in all fields simply due to the quantity of publications. Industry and research fields with long-running standards and practices[1] require especially high effort in result dissemination, to allow evolution to happen. Releasing either passive or interactive visualization can help people with varying technical backgrounds to understand even complicated results, either as supplementary to a written publication or on their own. Furthermore, publishing well-documented, open-sourced software packages allows new methodology to be adopted faster and to be easily built-upon. In return the software author can receive validation and improvement suggestions.

This section describes the main software and animation publications part of the dissemination efforts of the thesis research. In total two (2) software packages and eleven (11) animations are listed below. Originally the software and video publications were released in well-established, easily accessible online services GitHub and Youtube, respectively. The services allow the content to be updated and extended afterwards, and in addition they allow communication between the authors and the users. The animations have been used in numerous presentations by both the author's research group and by collaborators all over the world, prompting positive feedback.

Software

Below is a list of open-source software packages published as part of this dissertation. The source code is publicly available at the research groups GitHub page.[2] All of the listed software was written solely by the thesis author. The QR codes contain the respective GitHub repository web addresses of the software packages.

Software I Åkerblom, M. 2017, December 8. InverseTampere/qsm-fanni-matlab: Version 1.1.1 (Version v1.1.1). Zenodo. `http://doi.org/10.5281/zenodo.1098039`

The published software is the Matlab implementation of the Foliage and Needles Naïve Insertion algorithm, for generating leaf covers for a QSM. The leaf cover is defined by arbitrary leaf parameter distributions given by the user. Published under the GNU General Public License v3.0. Available in GitHub: `https://github.com/InverseTampere/qsm-fanni-matlab`

[1] For example, the field of forest science and forest industry in the case of the thesis research.

[2] InverseTampere GitHub, `https://www.github.com/InverseTampere`

Software II Åkerblom, M. 2017, December 11. InverseTampere/qsm-blender-addons: v0.6.0-alpha (Version v0.6.0-alpha). Zenodo. http://doi.org/10.5281/zenodo.1100793

The published software, written in Python, is an add-on for Blender for importing reconstructed tree models in various formats. The software allows users to easily create graphics and animations featuring QSM data, and to colour the model elements, e.g., based on branching order. Leaf model import is also possible. Published under the GNU General Public License v3.0. Available in GitHub: https://github.com/InverseTampere/qsm-blender-addons

Animations

The following animations were produced as part of this dissertation, listed in chronological order. All of the animations have been published at the research groups Youtube page.[3] The DOI is reported both in text and as QR codes, together with a link to view the video on Youtube. In total as of 2 May 2018 the animations have been viewed 9135 times on Youtube. All of the animations were produced by the thesis author, including design, animation and sound work. Some of the animations feature research data provided by collaborators.

Animation I Åkerblom, M. 2013. Creating surface patches. Zenodo. http://doi.org/10.5281/zenodo.1098718

A short demonstration of how surface patches/cover sets are created with the help of spherical environments centered at pseudo-random point cloud elements. Duration: 1 minute 8 seconds.
Available on Youtube: https://youtu.be/G1JkgtZjdXI

Animation II Åkerblom, M. 2013. Surface patch characteristics. Zenodo. http://doi.org/10.5281/zenodo.1098836

A visual explanation of using principal component analysis for surface patches/cover sets, in order to determine their geometric properties. Duration: 1 minute 11 seconds.
Available on Youtube: https://youtu.be/9yY5BTSGQxQ

[3] InverseTampere Youtube, https://www.youtube.com/user/TUTInverseProblems

Animation III Åkerblom, M. 2013. Locating the trunk and base. Zenodo.
http://doi.org/10.5281/zenodo.1098848

Animation of how the stem can be identified in a single tree point cloud, while excluding returns from the ground and undergrowth. The animation utilizes a real lidar point cloud. Duration: 1 minute 27 seconds.
Available on Youtube: https://youtu.be/d1PRDmzwsgQ

Animation IV Åkerblom, M. 2013. Automatic segmentation. Zenodo.
http://doi.org/10.5281/zenodo.1098874

Illustration of the segmentation sub-procedure of the TreeQSM reconstruction method, using an artificial tree surface model and lidar data. Colors are used to identify point cloud subsets essential to segmentation and bifurcation detection. Duration: 2 minutes.
Available on Youtube: https://youtu.be/PKHJQeXJEkU

Animation V Åkerblom, M. 2014. 3D Forest Information. Zenodo.
http://doi.org/10.5281/zenodo.1098882

An overview of the inputs and outputs of the TreeQSM method. The animation showcases how a multi-tree point cloud is transformed into a collection of structure models, and what attributes can be derived from them. The animation features data collected by E. Casella and tree models reconstructed by P. Raumonen. Duration: 2 minutes 6 seconds.
Available on Youtube: https://youtu.be/wANRdliE1zQ

Animation VI Åkerblom, M. 2014. 3D metsäinformaatio. Zenodo.
http://doi.org/10.5281/zenodo.1098895

Finnish version of ***Animation V****. Duration: 2 minutes 6 seconds.*
Available on Youtube: https://youtu.be/eS36t7ZdnBY

Animation VII Åkerblom, M. 2015. Cylinder reconstruction. Zenodo.
http://doi.org/10.5281/zenodo.1098898

The animation shows in detail how a segmented point cloud is reconstructed as a surface model by fitting several cylinders. The featured point cloud data is artificial but the reconstructed tree model was computed with the TreeQSM method. Duration: 4 minutes 41 seconds.
Available on Youtube: https://youtu.be/j0Emjwp-fmU

Animation VIII Åkerblom, M. 2015. Sylinterirekonstruktio. Zenodo.
http://doi.org/10.5281/zenodo.1098902

*Finnish version of **Animation VII**. Duration: 4 minutes 41 seconds.*
Available on Youtube: https://youtu.be/TeZYArQOrnw

Animation IX Åkerblom, M. 2015. Visualizing reconstructed tree models. Zenodo.
http://doi.org/10.5281/zenodo.1099523

An illustration of different visualization options for QSM data, using the same cylinder parameters and different textures. Visualization is done with both individual cylinders as well as lofted Bézier curves. Duration: 2 minutes 4 seconds.
Available on Youtube: https://youtu.be/pbpCrXb00s8

Animation X Åkerblom, M. 2016. Quantitative Structure Models. Zenodo.
http://doi.org/10.5281/zenodo.1099608

An overview of the TreeQSM method and the resulting models, compiled mainly of previous animations. Duration: 3 minutes.
Available on Youtube: https://youtu.be/A42Xg-6Sqkw

Animation XI Åkerblom, M. 2017. Tree species recognition with quantitative structure models. Zenodo.
http://doi.org/10.5281/zenodo.1099684

*Supporting material for **Article IV**, visualizing how the selected species classification features are defined and what levels of separation can be achieved with them. Duration: 9 minutes 32 seconds.*
Available on Youtube: https://youtu.be/SX6kYeuY0Oo

Introduction 1

Forests are a key part of the environment that surrounds us, enabling the conversion of carbon dioxide and water into oxygen, through photosynthesis. Furthermore, trees and plants are also essential for many of our planets animal species, as they offer nutrition and shelter. On the other hand, trees can be processed into wood and pulp that are the core of two major industries — especially in Finland — the lumber and paper industries. With the ecological and economical points-of-views often competing, it is important to measure the current state of the forests and to understand the dynamics of plants. As an example of what to measure, biomass information has been determined to be a key descriptor in the study of the global carbon cycle (Kaasalainen et al., 2015), and forest productivity and forest sequestration (Bi et al., 2004).

Before the 20th century, deforestation was happening in all climatic domains to make room for agriculture and urban development. While this is still the case in the tropical domain, deforestation has slowed or reforestation has begun in the temperate and boreal domains. On a global level the forest area was measured to be a little below 4 billion hectares — about a third of the total land area (FAO, 2016). Surprisingly even after centuries of research, many open questions remain regarding all aspects of forests, including the magnitude of forest resources, the role of forests in the carbon cycle, and the structure of individual trees and interaction between trees. For example the estimate of the total number of trees in the world was recently updated from the previous about 400 billion trees to 3 trillion (Crowther et al., 2015), furthermore recent study suggests that tropical forests might be net carbon sources rather than sinks due to deforestation and forest degradation (Baccini et al., 2017).

With climate change focusing research into the global aspects of forests, the local aspects can be overlooked. However in the end, understanding the phenomena globally comes down to studying the state and dynamics of single plants, competing with their neighbouring plants and interacting with their surrounding environment and soil. However, because of the massive area of the worlds forests and the lack of access to large parts of them, it is impossible to survey each tree even once, yet alone periodically to study their dynamics. Instead, much smaller areas are surveyed and the results are upscaled to receive calibration data and validation for measurements done via satellites for larger scales (Calders, 2015). Upscaling is an approximation with many sources of error, from the assumption of uniformity to the selection of the upscaling factor. Certainly, a key goal is to minimize the error associated with the values to be upscaled, here meaning the measurements from individual trees.

Tree architecture can be used together with process models to study the mass, energy and information exchange between plants and the environment. Furthermore, the architecture can be seen as a network for transferring nutrients and signals inside a plant (Godin and Sinoquet, 2005). Forest resource management, fire risk modelling and habitat mapping are just a few of the applications that rely on accurate tree models (Côté et al., 2011). Additionally, timber assortment, tree quality, branch decay time and carbon cycle estimations require detailed information about the branching structure of trees (**Article II**).

As an evolving, complex organism a tree proposes many difficulties for recording its state and dynamics. The first difficulty is biodiversity, as over 60 thousand tree species are known today (Beech et al., 2017), and thus methods for one species might not work for the next. In more tangible terms the size and shape of trees varies a lot, both between species and between trees of the same species. For example, the tallest recorded tree, the *Hyperion*, is 115 meters tall (Preston, 2006), and the largest trunk circumference is 36 meters for the *El árbol del Tule* (Debreczy and Rácz, 1997). Additional challenges are proposed by the intricate structure of a tree, consisting of the above-ground woody parts — the stem and branches — foliage or needles, and the stump and root system mostly below the ground surface. Branches, foliage and needles, all create occlusion for the tree and any neighbouring plant, while using any optical instrument. The obvious obstacle in studying tree roots is access. The options are destructive, *i.e.* felling the tree and pulling out the stump and root system (Liski et al., 2013; Smith et al., 2014), and non-destructive, using imaging technologies such as the ground-penetrating radar (GPR) to map the root architecture (Borden et al., 2017). To measure chemical content of leaves and branches, such as water or chlorophyll levels, manual sampling of standing trees has traditionally been the only available approach.

Recording tree architecture is also possible manually but it is extremely time-consuming, and in most cases carried out destructively (Dassot et al., 2011). Such measurements would have to include the measuring and documentation of branch lengths, diameters at certain intervals, branching points and angles. Because of the required man-hours, and the fact that often the goal is to estimate total tree volume — or to derive a biomass estimate based on the volume — it is common to use allometric equations to estimate total tree volume, from an easily measurable quantity, such as stem diameter at breast-height (DBH). The parameters of the equations are optimized using the available manually measured data. Although allometric models allow the estimation of various attributes at large scales, they are error-prone for many reasons, related to both the model and the measured data.

As most models, allometric equations are simplifications and approximations, thus resulting in uncertainty. Furthermore, most of the models are not universal, they do not account for species diversity or regional differences. Either a completely new model has to be developed, or at least the parameters have to be re-optimized when considering additional species or regions. However, a universal allometric model has been suggested by Chave et al. (2014). Additional error is introduced by the measurements associated with the optimization data and with the independent variable, *e.g.* DBH. Studies have shown that allometric equation derived above-ground biomass (AGB) estimates can have a relative uncertainty of 80 %, and thus the requirement for an alternative is high (Burt, 2017).

Light radar — lidar — or light detection and ranging (LiDAR) is a measurement technology based on either the time-of-flight, or the phase-shift, of a light impulse. Lidar scans can be performed terrestrially, either statically with a tripod or with mobile scanners (Kukko et al., 2012), airborne — from a drone, helicopter or a plane (Hyyppä et al., 2008) — or spaceborne from a satellite (Stysley et al., 2015). However, in this thesis only terrestrial laser scanning

(TLS) with a tripod is considered. With a TLS instrument a single measurement gives a range associated with the scanners azimuth and elevation, which can be transformed into a three-dimensional Cartesian coordinate system. Modern scanners can acquire up to a million range measurements per second with usually a 360 degree horizontal field-of-view and up to 320 degree vertical field-of-view. Phase-shift based scanners have a higher accuracy (1–3 mm at 25 m, max. range 200 m) but time-of-flight scanners have a higher maximum range (7–10 mm at 50 m, max. range $>$ 1000 m). Multiple scans from different directions can be *registered* to a uniform coordinate system using various techniques (Dassot et al., 2011). The collection of 3D elements resulting from a lidar scan is commonly referred to as a *point cloud*. Each element will have an intensity value attached to it, or multiple such values if a more advanced hyper-spectral lidar (HSL) — measuring intensities of multiple wavelength laser beams — is used (Chen et al., 2010).

In itself, a lidar scanner can directly only measure ranges — and the intensity values. However, when combined with data analysis techniques, they can be used to indirectly measure a lot more. Traditionally, lidar scanners have been used by engineers and researchers to measure and document external walls of buildings and archaeological sites (Dassot et al., 2011), but as early as in 2004 it was shown, that several tree attributes could be reliably extracted from TLS data (Hopkinson et al., 2004). The simple tree attributes included stem location, tree height, DBH, stem density and timber volume. A similar study was conducted by Thies and Spiecker (2004), but they only considered DBH, tree height and the starting height of the tree crown. Their results include a statement that the use of automated data acquiring and attributes derived from that data are more objective than those measured manually by a person or persons. Later Olofsson et al. (2014) successfully used the random sample consensus (RANSAC) algorithm to record stem count, DBH and tree height. Kankare et al. (2014) used a combination of TLS data and field measurements to study tree quality.

There are several sources of error and uncertainty when trying to comprehensively lidar scan a tree, from either one or more directions. The measurement density is not constant; Coverage becomes sparser when moving towards the tree top, *i.e.*, farther from the scanner. Furthermore, environmental factors such as wind, fog or rain can affect the scan quality, as can either self-occlusion or occlusion by other plants. Additional error can be introduced also in the registration of several scans (Côté et al., 2011).

It is important to note, that the early applications of lidar instruments in forests, aimed to measure the same forest inventory parameters as had been measured for decades by-hand. Only later methods were developed to reconstruct more comprehensive descriptions of the tree structure. A detailed survey on the development of such methods will be presented in Sect. 1.1. The advancement in the area has been truly impressive as in 1999 it was stated that, with the current technology it is not possible to automatically acquire neither the topology nor spatial coordinates of plant components (Godin et al., 1999).

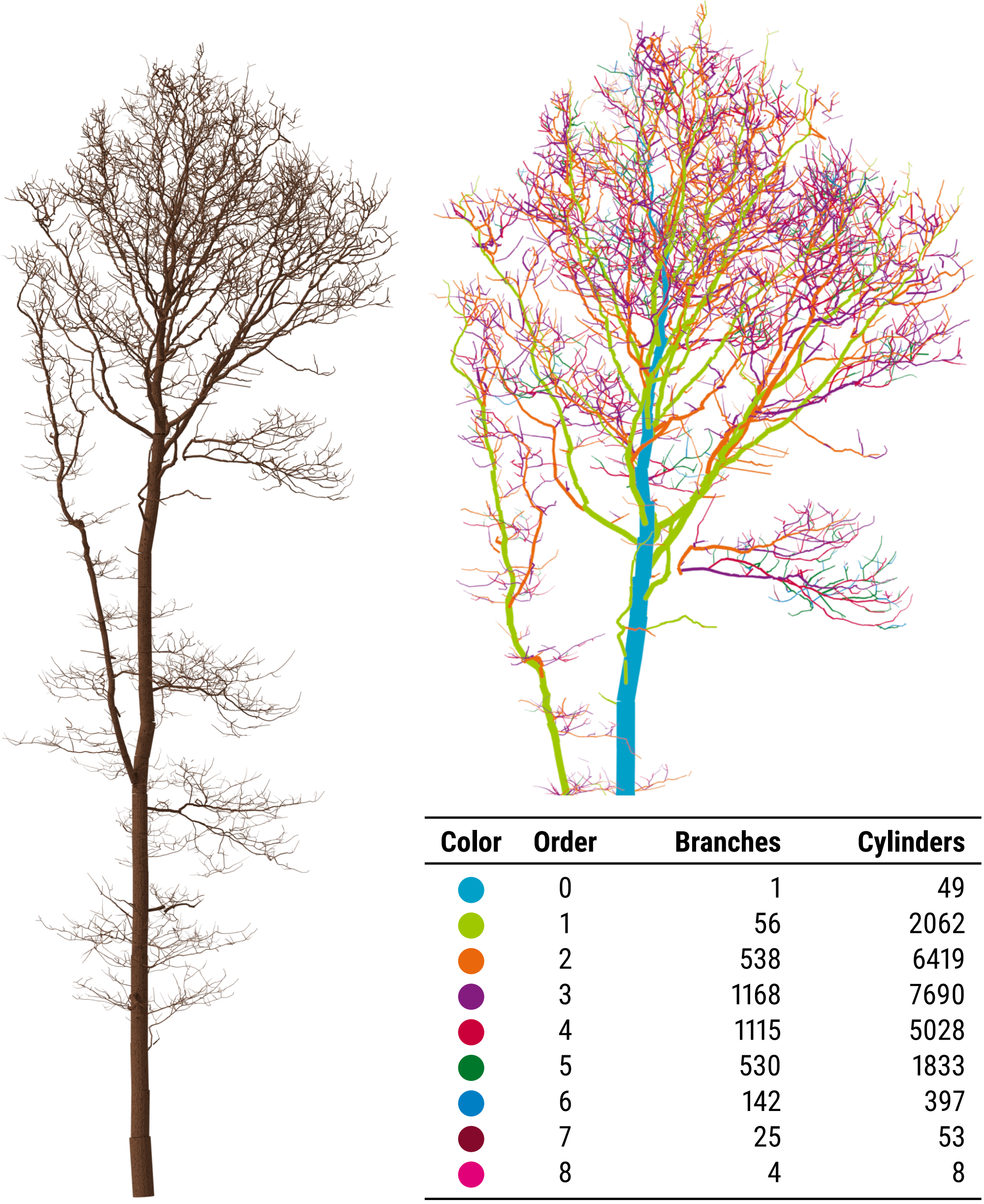
The hypothesis is that if a comprehensive tree architecture–surface model can be reconstructed from TLS data, it then provides access to a variety of tree attributes previously unavailable or extremely hard to measure. Nonetheless, the model still provides access to the more traditional measures, like tree height and DBH should applications still require them. The new attributes include, *e.g.*, the branching topology, branching angles, partial and total volumes, but are certainly not limited to these. One of the key differences to previous field measurements, is that if a comprehensive structure model is compact enough to store, it allows the computation of additional values or distributions at any given time.

At the core of this thesis is a novel method — TreeQSM — for reconstructing a comprehensive quantitative structure model (QSM) of a single tree from TLS data, designed to fulfil the above hypothesis, providing access to compute quantitative tree attributes. Although similar approaches for reconstructing tree architecture and/or the tree surface have been presented before and after TreeQSM, only a few match the accuracy, speed and comprehensiveness of the reconstructions, and none have been as widely accepted for operational use. As such, it is not meant for measuring a specific attribute, but rather any structural parameter currently in use, or even to be defined in the future. The TreeQSM implementation offers a simple, easy to approach user interface (UI), which has allowed the method to spread fast and to be accepted by researchers, working on forest and ecosystem research, as a valuable tool. The TreeQSM method is presented in **Article II**, with additional validation in **Article I**.

The TreeQSM method transforms the input point cloud data into a QSM, consisting of cylinders, or other geometric primitives, providing a compact format for storing and disseminating tree information. An example of a cylindrical QSM is shown in Fig. 1, coloured in two different ways demonstrating the branching topology contained in the model. Although a circular cylinder is most commonly used as the geometric primitive with TreeQSM, it is not the only option. Thus, the differences between several options are studied in **Article III**. The surface reconstruction procedure of the TreeQSM method is visualized in **Animation VII**/**VIII**, using circular cylinders. The resulting QSM format offers easy access to both geometric and topological tree attributes. Furthermore, the compact format allows studying the tree structure evolving over time (Kaasalainen et al., 2014).

In addition to the method itself, the thesis focuses on three aspects of the TreeQSM method: validation, applications and dissemination. The former is perhaps the most fundamental, as validation is required to show that any method works as described. Early validation, mainly in terms of volume reconstruction accuracy, is presented in **Article I** and **Article II**. Furthermore, several validation studies carried out since the publishing of the TreeQSM method, are presented in Sect. 2.6. The studies consider the accuracy of TreeQSM in regard to commonly-used tree properties, such as, tree volume, DBH and AGB. Error related to the selection of the geometric primitive, that a QSM consists of, is studied in **Article III**.

Included in the thesis are two applications of reconstructed QSMs, the first of which is species recognition. Having accurate species information is key in forest management applications (Puttonen et al., 2010), and tree species is one of the few tree parameters that are recorded during manual forest inventories. Rather than done manually by experts, **Article IV** shows that automated species recognition is possible in larger scales, by utilizing QSMs. The process involves selecting several classification features, computed from the reconstructed tree models, that have sufficient levels of variation between different tree species. **Animation XI** was produced to augment the written publication, by visualizing the



Color	Order	Branches	Cylinders
🔵	0	1	49
🟢	1	56	2062
🟠	2	538	6419
🟣	3	1168	7690
🔴	4	1115	5028
🟢	5	530	1833
🔵	6	142	397
🟤	7	25	53
🔴	8	4	8

Figure 1.1: An example reconstructed QSM of an oak tree, visualized as a collection of cylinders. Left: Full tree with a bark texture. Right: Top of the tree with each branch order coloured with separate colours, listed in the table with branch and cylinders counts.

definition of the included classification features, and to demonstrate the resulting levels of species separation for each of the features.

The other included application is leaf cover generation on the structure defined by a QSM. Leaves and needles are responsible for tree photosynthesis and they largely define how a tree intercepts light (Casella and Sinoquet, 2007). Thus, foliage and needles are key in studies dealing with atmospheric interactions, and how much light passes through forest canopies. Unfortunately, there currently exists no software able to reconstruct individual leaves from TLS data. As an alternative to reconstructing leaves, **Article V** presents a method for generating them based on arbitrary, user-defined leaf parameter distributions, to populate a QSM with a leaf cover. The source-code of the MATLAB implementation of the proposed method, **Software I**, is published online.

In addition to the applications presented in this thesis, it would be easy to list numerous current and future applications for forest reconstruction. Especially in terms of machine learning high-dimensional QSM based data is ideal, as demonstrated with the species recognition application. Tree and forest structures are too complicated to analyse comprehensively by the human eye. However, accounting for each branch and bifurcation is possible for an automated system, for example in a computer-assisted harvester system. Such systems have already been developed (FIBIC, 2014), and could be used to guide the harvester operator helping maximise productivity while minimising tree and soil damage.

The final aspect of the thesis is the dissemination of the TreeQSM method and the advances it enables in multiple fields. Printed publications and static images may still be the most common choice for scientists, but they are certainly not the only option (Perkel, 2018). Especially due to the complex nature the reconstruction approach and the long traditions in the field of forest science and the forest industry, numerous ways of dissemination were utilized, to explain the details and capabilities of TreeQSM. Several animations were produced to visualize key details of the reconstruction method (**Animations I–III**) and overviews of the complete method (**Animations V**/**VI**, **VII**/**VIII**, **X**). Furthermore, reconstructed QSMs were disseminated as interactive, web-based 3D models, both on their own and with attached tree property and distribution data. The extensive dissemination has accelerated the process of adopting the use of the TreeQSM method in operational use.

Animation IX demonstrates the options available for visualizing QSMs as collections of cylinders or as branch-level continuous surfaces, with or without textures. **Software II** is a tool to import cylinder data from a QSM into a 3D graphics program to produce images and animations similar to the ones part of this thesis.

1.1 Related research on tree architecture reconstruction

Many approaches for extracting tree architecture have been presented in the past. Some of them aim to reconstruct only the branching architecture as a graph, others evaluate only attributes as volume, and some reconstruct stem and branch surfaces. There is a lot of variation in the level of detail of the reconstructions, the primary applications, level of automation and required computational resources. Clear differences also occur in the design with respect to the input data; *Data-driven* methods only reconstruct parts well represented in the data, and *model-driven* approaches augment the data with, for example, biological growth models, to create geometry or graph nodes with little to no measurements.

A majority of the existing methods for reconstructing trees from TLS data are presented below, categorized by their primary outputs, *i.e.*, whether the methods have been developed to reconstruct the branching graph, total or partial tree volume, or the tree surface. Note that some methods reconstruct both a branching graph and a surface model, and thus can be listed in multiple categories. The section ends by presenting additional methods for reconstructing tree models from photographs.

1.1.1 Reconstructing tree skeletons

A tree skeleton is a graph describing the branching topology of a tree. Typically, the nodes of the graph are located inside the tree geometry and at least each branching point should have a single node, although, additional nodes can be used to describe the geometry of the branches in more detail. The edges of the graph follow the tree topology and geometry, and the edges usually flow through the core of the stem and the branches. Multiple approaches have been presented for reconstructing a tree skeleton from terrestrial lidar data, with various levels of included detail and automation.

Cheng et al. (2007) presented a method for reconstructing a tree from a single range image obtained by a laser-scanner. Their method operates on a two-dimensional image plane and performs skeletonization based on the difference of ranges between neighbouring pixels. User input is required to connect parts of the skeleton. Their method is naturally not able to reconstruct branch parts that are hidden from the viewing direction.

In the method proposed by Xu et al. (2007) a point cloud from a single or several registered laser-scans is processed to produce first a plausible skeleton and finally a corresponding surface mesh. Their algorithm is developed for computer graphics applications, and therefore, does not aim for a strictly faithful reconstruction but instead a plausible result with a realistic appearance.

Dijkstra's shortest path algorithm is used to form an initial graph. The points are then clustered into bins and skeleton nodes are computed as centroids of each bin. Possible sub-graphs are connected to the main skeleton using bi-directional probing. Locations of leaves are estimated from the point cloud and missing support branches are synthesized for them by copying and scaling existing sub-graphs. Their method was criticized for unfaithful reconstruction of the branching structure and for unrealistic loops in the graph structure (Yan et al., 2009).

A very similar approach is suggested by Côté et al. (2009) as both methods reconstruct the main skeleton with Dijkstra's shortest path algorithm and use the points reflected from foliage as attractors to *grow* plausible branches to overcome the shortcomings of the point cloud coverage.

(Yan et al., 2009) present a novel method for tree skeleton extraction. Their method starts by selecting a certain number of clusters defined by a center point and its k nearest neighbours. The direction and radius of each cluster is estimated from the point cloud and a bounding cylinder is fitted to each cluster. The cylinders are not used to represent the tree surface, but simply to measure the correctness of the clustering. If a cylinder fits well to the data, the cluster is accepted, and otherwise it is divided into two sub-clusters and the process continues iteratively.

Where (Xu et al., 2007) only used the centroids of the clusters, Yan et al. (2009) use additional *junction nodes* placed at the boundary of neighbouring clusters and leaf nodes to extract the tree skeleton using, again, the shortest path approach. The use of the junction points gives more accurate node distance measurements. When compared to the very similar approach by Xu et al. (2007) the proposed algorithm works much better and does a clearly more faithful reconstruction of the tree skeleton.

As an alternative approach and an improvement to many of the earlier methods, Livny et al. (2010) propose a fully automatic, parameter free skeletal reconstruction method based on global optimization. The use of global and linear optimization makes their approach robust and fast. Furthermore, their input point cloud can contain multiple trees which can even overlap. They describe a process that is based on *branch-structure graphs* which are iteratively optimized with just a few general constraints.

The initial graphs are formed using Dijkstra's shortest path algorithm on a point cloud where the tree bases have been automatically located. The graphs are then iteratively optimized based on assigned node weights and constructed orientation fields. The optimization typically converges after just a few iterations.

Preuksakarn et al. (2010) also presented a similar approach to reconstruct manually extracted single-tree point clouds as generalized cylinders. The procedure begins by *contracting* point cloud elements toward the center of their respective branch, that is defined as the centroid of a spherical neighbourhood. The skeleton of the tree is reconstructed as a graph by using the contracted points as attractors, and placing nodes so that the attractors are visited. Original point cloud elements are assigned to nearest node, and projected in the respective skeleton direction to estimate the branch radius. The method was tested by comparing the geometry and topology of an apple tree, reconstructed with the approach and with a digitizer by an expert. The results showed a 90 % match.

Bucksch et al. (2010) presented a skeletonization algorithm based on dividing the containing space into an octree structure. Although, the method was not designed solely for trees, it is demonstrated to work with TLS data from trees. In the algorithm each octree cell containing lidar returns was placed with an initial octree graph node at the mass center of the respective point cloud elements. Nodes in adjacent cells are connected by edges, defining the octree graph. The final tree skeleton is received by a reduction procedure, which is the most time consuming part of the process. The graph reduction was completed in 192 minutes for an apple tree and in about 275 minutes for a tulip tree, on a 2.66 GHz dual-core processor.

The (Côté et al., 2009) method was perfected by Côté et al. (2011) and named *L-architect* (lidar data to tree architecture). The trunk and branch reconstruction process remains identical to the previous version, but the foliage appending is introduced as a global optimization problem in order to make the method more automatic. Several additional *in situ* measurements are required to define the objective function which is then optimized using the

Global Optimization by Multilevel Coordinate Search algorithm. The required measurements are DBH, total foliage area, and material distribution in the vertical direction.

1.1.2 Tree volume reconstruction

Not all tree reconstruction approaches aim to recording the topology of a tree. In certain applications it is key to estimate only the volume of a tree, the stem or the branches, and *voxelization* is one way to receive those estimates. When voxelizing the space containing a tree, the space is divided into, usually cubical, voxels with a fixed edge length. The goal of the volume reconstruction is to determine the voxels that are lay inside the tree hull and make up the tree. The level of detail in voxel-based models is dependent on the voxel size. *I.e.*, large voxel sizes result in lower spatial resolution, and with the decrease in voxel size, the computational requirements grow fast.

Gorte and Pfeifer (2004) presented a method for finding the skeleton structure of a tree from a point cloud. Their approach uses voxelization and mathematical morphology techniques, and produces as an output a point cloud segmented into individual branches. Similar to (e.g. Xu et al., 2007), Dijkstra's shortest path algorithm is used when segmenting the acquired tree skeleton, but now it is applied in the voxel space rather than on the point cloud itself.

A couple years later, Lefsky and McHale (2008) suggested a very similar approach and tested their method extensively on a couple of hundred urban trees with eleven different tree species, and an average of 640000 points per tree. Their method does not directly aim to reconstruct the branching structure, but rather to estimate the stem and canopy volumes. To validate the method over 250 diameter field measurements were made with a dendrometer, and were found to be highly correlated with the corresponding diameters received from the lidar measurements.

The proposed method begins by considering only voxels that have lidar returns in them, and by selecting the stem base voxels manually. Next, *stem sections* are identified with a three-dimensional search algorithm and the 26-connectivity of the voxels. These sections form *layers* of the stem and later neighbouring, connected layers are merged iteratively. The stem volume was approximated by fitting cylinders to the original points forming the final stem sections. Furthermore, the canopy volume was estimated based on the number of lidar returns.

The proposed algorithm gives good results for the stem and canopy volumes, but requires a lot of computational resources and time for manual editing of the stem sections, especially at branching points. The authors also state that the simple voxel neighbour search does not work very well and causes the stem sections to become non-normal to the stem direction, which in turn causes erroneous volume estimates. The authors also suggest using a cone rather than a cylinder in the volume computations.

Vonderach et al. (2012) propose a voxel-based approach for acquiring branch volume, DBH and tree height. Their simple method uses horizontal layers and several 2D approaches to identify the interior voxels. The number of interior voxels is used to find an estimate of the specified tree properties. The method is designed for an urban environment and high-resolution scans consisting of 20–60 million points for each tree. Tests on nine deciduous trees showed good results with about 0.5 to 1 meter over estimation in tree height and only a few tenths of a millimeter in the DBH. The computational times are not discussed but the authors mention a *high effort in analysis*.

Similarly, Bienert et al. (2014) use vertical slicing to determine both exterior and interior voxels. In a effort to account for volume overestimation, the volume of the exterior voxels is estimated by the volume of the smallest axis-oriented bounding box containing the voxel points. Interior voxels are unlikely to have lidar returns, and their volume is thus computed the normal way. The accuracy of the method was tested on 13 young, short maple trees, that were scanned leaf-off with from positions without occlusion. Reference volume measurements were performed by immersing tree segments in a water tank and measuring displacement. The result showed that applying the proposed volume correction lowered the average relative volume error for the tree trunk volume from 74.2 % to just -0.3 % and the root-mean-square error (RMSE) from 161.2 % to 11.6 %, compared to the non-corrected voxel-based volume estimate.

1.1.3 Surface reconstruction

Where in voxel-based methods the tree was presented on a fixed cubical grid, in surface reconstruction the surface of the stem or branches are presented by either a single parametrized surface or a collection of such surfaces. Surface models can also be used to estimate the tree volume, but also other properties, such as, local diameter or direction of a branch.

As a continuation to (Gorte and Pfeifer, 2004), in an article published the same year, Pfeifer et al. (2004) suggest using circular cylinder fitting on the segmented point cloud to get an approximation of the tree surface and volume. Starting from the base of a segment a cylinder is fitted as a least-squares fitting problem to the point cloud data associated with that part of the segment. Local geometric properties of the point cloud are used as initial values. If the fitting is successful, the cylinder hull is moved in its axis direction for a certain small distance, and the points close to the updated cylinder position are used to fit a new cylinder. The *cylinder following* is repeated for each segment until a fitting failure occurs or all points in the segment are processed. However, the method proposed by Pfeifer et al. (2004) is only able to reconstruct small parts of the thickest branches and the tree stem. The resulting cylinder model is incomplete and disconnected.

Cylinder fitting was also used by Cheng et al. (2007) to reconstruct the surface around the skeleton by using geometric properties of point sets as initial values. Regarding only using a single lidarscan, the authors propose future work, where several range images would be used, such that a cylinder model would be computed for each direction separately and later combined to a single model. However, no such work has since been published.

In (Xu et al., 2007) the surface was reconstructed around the skeleton graph by estimating branch diameter at each skeleton node. Allometric theories, such as, Murray's law (Murray, 1926) and the refined pipe model by West et al. (1999), are used to estimate branch thickness. The circles defined by the diameter estimates could be connected with either triangles or smooth spline surfaces. The authors claim the method to be fast and quite automatic. However, they do admit that the tuning of overall 7 parameters requires some expertise. Furthermore, they state that the utilized allometric theories are not able to account, *e.g.*, for the difference in the branching structures of old trees, compared to younger trees of the same species.

Quite similarly, Yan et al. (2009) transform the completed tree skeleton into a set of B-spline curves, as a process called *lofting* (Gomes et al., 2012, p. 230) is used to get a continuous surface presentation of each branch with the help of the radius estimates given by two-dimensional circle fitting.

The method by Yan et al. is described to be fully automatic but still there are several parameters to account for, such as, the initial number of clusters and the stopping conditions for the iterative processes. With a point cloud of 217 000 points the time required for segmentation is 3 minutes and 35 seconds, and the total time including branch modelling about 5.5 minutes. The computer specifications are undefined.

Côté et al. (2009) used the original pipe model (Shinozaki et al., 1964a,b) to get an approximation of the tree surface around the reconstructed skeleton. On top of that, foliage shoots were appended to the tree geometry. The reconstruction approach was validated by comparing first and second order reconstructions: laser scanning was simulated on the first order reconstruction and second order models were reconstructed. The results show excellent reconstruction quality in their test set of four coniferous trees, evaluated using the crown density, leaf area, and area of the woody structure. The method has several parameters that are tuned interactively based on visual appearance, and thus it is far from an automated solution.

The results given by the L-architect method, in (Côté et al., 2011), are convincing as, *e.g.*, the root mean square error of the vertical material distribution in the five test cases are quite low. Furthermore, the rendered visualizations are certainly impressive with the included bark textures and included foliage. A clear downside of the method is the computational time which can be anything from 2 to 31 hours on over 5000 processors each operating at 2.8 GHz. Other than that, the approach performs very well and delivers on the promise of detailed reconstructions even when wind and target occlusion are present.

Livny et al. (2010) generate the surface geometry using global optimization, similarly as with the skeleton reconstruction. The respective sizes of the sub-graph of the nodes are used as weights in the optimization procedure. L-systems are used to add fine branches to the model. Example point clouds were collected with a continuous scan made by a scanner mounted on a vehicle, travelling at a normal driving speed in an urban environment. *E.g.*, a scan containing approximately 300 000 points and 5 trees, took 2 minutes to process on a computer with a 2.7 GHz processor.

In a comparison study, the methods by Xu et al., Livny et al. and Preuksakarn et al. were compared to reference structures recorded by expert researchers in a study by Boudon et al. (2014). The results showed that in the case of the four test trees, volume was underestimated by 7 % on average, while total branch length was overestimated between 15-70 %, depending on the method. After determining geometric and structural similarity indices, the authors conclude that the methods by Xu et al. and Preuksakarn et al. work better that the method by Livny et al..

Rather than using skeletonization, Liang et al. (2012) compute flatness values and normal vectors for each single-scan point cloud element, by analysing the geometric properties of the point and its k nearest neighbours. By filtering points with these properties, potential stem returns are identified and grouped according to distance. The groups are reconstructed as consecutive cylinders from base towards the top, as far as sufficient data is available.

The M-estimator Sample Consensus algorithm is used to detect individual tree stems from multi-position scans by (Kelbe et al., 2013). The detected stems are divided into regions by taking vertical slices, and each region is reconstructed as a cylinder, by combining line and circle fitting. Post-processing steps are taken to ensure that consecutive cylinders do not differ too much in direction and radius, by removing outlier cylinders and using spline fitting to find the stem axis. Linear interpolation is used to find a smooth taper curve.

Hackenberg et al. (2014) present a method for reconstructing cylinder-based models from manually isolated single-tree point clouds. The process starts from the base of the stem, which has been detected by extracting the lowest slice of the point cloud. A circle is fitted to the points in this slice to find the starting point of the first cylinder. A sphere with a radius slightly larger than the circle is placed at the starting point. The point cloud elements close to the sphere surface are isolated and circles are fitted to the sufficiently large components. Cylinders are appended to the resulting model to connect the previous starting point and the center points of the new circles. The process continues iteratively for each of the new circles, processing the largest first. Post-processing steps are used to fix branching junctions, and to record the branching topology.

The method is tested with both real and artificial point clouds and reconstruction accuracy and level-of-detail are high. The authors note that the method still lacks the detection of incorrectly added cylinders, and may thus require manual corrections. However, the method was further developed in (Hackenberg et al., 2015a) to address the problem, using an allometric correction scheme.

1.1.4 Photo-based approaches

All the methods presented above, operated on lidar data, but that is not the only data source available for tree reconstruction. Photographs are one alternative even though they do not contain range information. Using multiple photographs taken from different angles it is possible to reconstruct objects (e.g. Neubert et al., 2007; Teng et al., 2007). The level-of-detail of the reconstructions is usually lower, but the motivation for using photographs is the low cost of the equipment, tine required for photographing, compared to operating a laser scanner. Furthermore, as the models generated from photographs are not usually applied in forest science applications, but rather in visualization applications, a lower level of resemblance between the input and the output trees is tolerated.

Shlyakhter et al. (2001) presented a method for reconstructing the main branching structure of a tree from several photographs. The received tree skeleton is then used together with a species-specific L-system to find a plausible fine branch structure and add foliage to the branches. The tree in each image is segmented from the background and the tree silhouette is used to construct a volumetric intersection which defines a visual hull of the tree. Voronoi nodes are then used to find a medial axis which is used as the main skeleton. The method is not fully automatic as the image segmentation is done manually. Usually, 7 to 14 images are used for a single tree, and the overall time required for a reconstruction is over four hours.

Teng et al. (2007) propose a very different approach in terms of image count as their method only utilizes two photographs. The algorithm is designed to reconstruct the tree stem and main branches from images, taken from slightly different positions. The use of only two images makes the process fast, due to reduced complexity of correspondence matching, but can result in inadequate 3D information and either partial or total occlusion of branches.

The process starts by segmenting the stem from one of the images interactively, and by extracting the skeleton of the selection. 2D stem points are selected from the skeleton, to form the graph of the stem. The camera calibration information and the second image are then used to trace the location of the 2D stem points in the third dimension. Generalized cylinders are placed on the 3D skeleton to form the final stem model. The total run time of the algorithm is not reported, but the authors state that the time is dominated by the manual segmentation which can take minutes. The resulting stem models are visually impressive, but the method does have limitations with occlusion.

Neubert et al. (2007) propose a novel way of producing approximate tree models, using particle flow simulation and two or more images. As their models are approximate and non-deterministic, strict registration of the photographs is not a requirement. As with the other photo-based methods the tree is first separated semi-automatically from the background of all the images. The main branching structure is then sketched on the extracted tree silhouette either by hand, which gives better results, or automatically. The main branching structure is used to first create a two-dimensional attractor graph for each plane. The graphs are then connected as a single three-dimensional vector field.

A voxel model is computed for the tree by approximating the density of each voxel from the input images. The density distribution is used to select the starting voxels of particles for the simulation; particle position inside a voxel is selected randomly and the attractor field is used to guide the particles during the simulation. The amount of initial particles is chosen based on the tree size. The particle simulation produces a three-dimensional graph of particle traces. On each point on the graph particle trace count and botanical rules are used to approximate the thickness of the branch at that point. Finally, leaves and tiny twigs are appended according to the density distribution to the tree model that takes only seconds to generate on a computer with a 3 GHz processor.

The method produces natural looking results that are similar to the input images, but the authors do state that their method does not work well with all tree species. Furthermore, particle flow simulation is not able to capture the smallest details accurately.

1.2 Objectives of the thesis

This thesis aims to answer the following research questions. Publications related to each question are given in parenthesis at the end of the list item.

1. Can the branch size distribution of a single tree be estimated accurately from a terrestrial laser scanning point cloud? What other tree properties can be computed from the models? Is comprehensive modelling possible?
(**Articles I – II**, **Animations I – VIII**, **X**)

2. What is the magnitude of the model-based error associated with the QSM reconstructions, and especially with the choice of the commonly used circular cylinder as the geometric primitive? (**Article III**)

3. Is some of the information contained in reconstructed structure models species-specific? And if so, is there enough of it to automatically recognise the species of a tree? (**Article IV**, **Animation XI**)

4. Can the structure models be augmented with a set of realistically distributed leaves for simulation and visualization applications? (**Article V**, **Software I**)

5. What different ways are available for visualizing and disseminating structure models and other related research results? (**Animations IX – X**, **Software II**)

1.3 Thesis outline

This thesis is divided into seven chapters. In Chapter 1 the context of the research is presented shortly, followed by a longer review of related research in Sect. 1.1, mainly focused on tree reconstruction from various data sources.

The novel TreeQSM reconstruction method is presented in Chapter 2, with the main steps of the algorithm described in Sects. 2.1–2.5. Studies validating the presented method are presented in Sect. 2.6. Chapter 3 shows how species-specific properties can be computed from reconstructed tree models, allowing automatic species recognition. Reconstructed structure models can be populated with a generated leaf cover with a method presented in Chapter 4. The chapter also describes how tree models can be used to describe and visualize distributions related to the tree. Chapter 5 presents approaches that have been used to disseminate tree models reconstructed with the TreeQSM method. Sect. 5.1 describes a method for rendering point cloud data, used in various animations part of this thesis. Different ways of visualizing the tree models are presented in Sect. 5.2. Details of all the animations included in the thesis are presented in Sect. 5.3, including what visualization methods were used for both the point cloud data and tree models. Interactive 3D models of reconstructed trees and a related software demonstration are discussed in Sect. 5.

The main results of this thesis are discussed in Chapter 6, where speculations are given on the future of forest inventories and virtual reality applications, in the light of advancements in tree and forest reconstruction. Conclusion of the results and discussion are made in Chapter 7, with directions for future work presented in Sect. 7.1.

TreeQSM – Tree architecture reconstruction method 2

In 2011 our team developed the first version of a method to reconstruct a single tree from a TLS point cloud, later to be called *TreeQSM*. The method was used as a tool to show that it is possible to approximate tree volume and branch size distribution from TLS data (Raumonen et al., 2011). After some improvements, such as better branch segmentation and the *gap filling* procedure, further validation was carried out in **Article I** and in more extensively in (Åkerblom, 2012). A more detailed explanation of the procedure for reconstructing individual trees was published in **Article II**. Further development, including adding segmentation correction, was introduced in Calders et al. (2015b), together with validation with a large number of real trees. Later the method was augmented with a tree extraction pre-processing step, allowing the input point cloud to contain multiple trees, showing that automatic forest-plot-level reconstruction is possible (Raumonen et al., 2015). An overview of the reconstruction algorithm is given next, but one can also be found in **Animation X**. For extensive details, refer to **Article II** and Åkerblom (2012). Recently the full source-code of TreeQSM was released in GitHub[1] while the development continues.

What happens between the TreeQSM input and output data is described, in order, in Sects. 2.1–2.5. The differences between the initial and current versions are also highlighted in the respective sections. All of the TreeQSM validation studies are presented in Sect. 2.6.

2.1 Input data and parameters

The main input of the reconstruction approach is a 3D point cloud of a single tree, that can be a combination of separate co-registered terrestrial laser scans, and that can contain additional returns from the ground and surrounding vegetation. Each of the returns consists of three spatial coordinates and either a single intensity value or numerous intensity values, in the case of a HSL. However, the intensity values are not utilized by the TreeQSM method as it relies only on geometry. Nonetheless, measurement intensities and recorded colour information may be useful in other data processing for, *e.g.*, wood–leaf separation and detecting leaf cover parameters.

Initial filtering is carried out for the point cloud data to discard lidar returns that are identified as not part of the tree, by being too isolated, and thus shouldn't contribute to the reconstruction and the resulting model. Such point cloud elements can also be caused by the measurement technology, in the form of phantom returns, or by self-occlusion or occlusion by other trees. An important goal of the filtering process is to keep all the returns from the tree, to ensure a detailed reconstruction.

[1] TreeQSM: `https://github.com/InverseTampere/TreeQSM`

2.2 Forming surface patches

The filtered point cloud P is partitioned into *surface patches* (also called *cover sets* in some of the publications) that conform to the tree surface, using an almost uniform Voronoi tessellation (Okabe et al., 2009). During the partitioning process center points, or *seed points*, $\mathbf{c}_i \in P$ are selected iteratively and randomly, so that

$$\| \mathbf{c}_i - \mathbf{c}_j \| > d_{\mathrm{min}}, \quad \forall\, \mathbf{c}_i, \mathbf{c}_j \in S,\ i \neq j, \tag{2.1}$$

where S is the set of selected seed points and d_{min} is a user-defined parameter. On the first segmentation run d_{min} is a fixed value, parameter PatchDiam1, but on the second segmentation run the distance limits vary locally, depending on two parameters PatchDiam2Min and PatchDiam2Max. To ensure the selection of seed points is comprehensive, the maximum distance between any point cloud element and the closest seed point is limited:

$$\forall\, \mathbf{x} \in P \;\; \exists\, \mathbf{c}_i \in S : \quad \| \mathbf{x} - \mathbf{c}_i \| < d_{\mathrm{min}} \tag{2.2}$$

The center points help define spherical environments R_i consisting of point cloud elements inside a sphere with a r_{i} radius:

$$R_i = \{\, \mathbf{x} \in P \;|\; \| \mathbf{x} - \mathbf{c}_i \| \leq r_i \,\}\,. \tag{2.3}$$

Depending on the segmentation run, the radii r_i can be a single fixed value, or they may vary between values by default derived from the PatchDiam2Min and PatchDiam2Max parameters.

Overlapping spherical environments, determined by shared point cloud elements, define a neighbouring relation for the environments and the resulting surface patches. To increase the likelihood of overlap, the parameters are typically chosen, so that $d_{\mathrm{min}} < r_i$. Elements part of any overlap are assigned to the environment with the closest center point, forming the final surface patches that are a partition of the initial point cloud. The procedure is visualized in **Animation I**, although it uses the alternative *cover set* term.

As a surface patch is a subset of the point cloud, principal component analysis (PCA) (Jolliffe, 2002) can be used to analyse its geometry, in order to later find patches with certain characteristics. The neighbouring surface patches and their principal components can also be used to derive characteristics for any given patch. **Animation II** visualizes how geometric variations in point clouds show up in the principal components, and what characteristics can be computed utilizing the principal components. In the reconstruction process mainly the direction, normal and dimensionality — surface patch in a small branch will be elongated (1D) while planar (2D) in the stem — are utilized to identify the stem in the point cloud.

Surface patches characterize the local geometric and topological properties of the tree. *E.g.*, a given patch is elongated and connected, as defined by the neighbour relation, to a set of other surface patches. Utilizing a *local-to-global* approach, these local properties can be used to determine global properties of the tree geometry and topology, through sub-procedures called segmentation and cylinder reconstruction, described in the following sections.

2.3 Branch segmentation

Even though the point cloud has now been partitioned into surface patches, there is still no direct information of the topology of the tree, or the geometry of its parts. In order to reconstruct the branching topology collections of surface patches, forming individual branches and the stem, have to be identified. The sub-procedure responsible for reconstructing the topology, is called *segmentation*. The original version of the procedure is presented in Sect. 2.3.1. The current, updated version, where segmentation is carried out twice, is presented in Sect. 2.3.2.

2.3.1 Original segmentation

Surface patch characteristics are used to find parts of the point cloud that are both planar and whose direction is parallel to the stem direction, *i.e.* vertical. After this simple filtering parts not belonging to the stem might still be included. Thus, only the largest connected component, in terms of point count, is selected as the initial stem set. This set can be augmented with components directly below or above it. The base of the stem is located by fitting a cylinder to the base of the current stem set. Surface patches are then added iteratively below the stem base (and removed above), and the cylinder fitting is repeated. The process is repeated while the ratio of two consecutive cylinder radii does not increase too fast. The final stem base is found when the radius increases very fast, with the inclusion of ground points.

Next, returns from the ground and surrounding vegetation are excluded by selecting neighbouring surface patches of the stem base set, without moving into the stem set. As this process is continued iteratively, all returns not part of the tree are included in the ground set that will be excluded from any further analysis. At this stage the stem and ground sets have been defined, and all the remaining surface patches are assigned into the branches set. The process of separating the stem, branches and the ground is described visually in **Animation III**. If the point cloud data are such that no non-tree returns are present, the base of the stem can simply be defined as the lowest *layers* of surface patches in the whole data.

Starting from the base of the stem set, the remaining surface patches are segmented into individual branches. The first layer of neighbouring surface patches form the initial *cut set*, that is updated with a new layer of neighbours at each step, while the oldest layer is appended to the current segment. While the cut set remains a single component the process repeats. However if the cut set becomes disconnected each component is analysed separately, and determined which components are a part of the segment. In the case of comprehensive data, the only reason why the cut set would become disconnected, is the presence of a bifurcation, meaning that some cut set components are located in separate branches. In this case, the smaller components of the cut set are labelled as bases of new segments, and extension of the current segment into those surface patches and their unvisited neighbours is prevented, while the larger component is assigned as part of the current segment. However due to imperfections in the data, cut set disconnections can also happen even though the components are part of the same branch. To counter this effect the cut set is actually extended a few layers further on each step, to see if the components become reconnected.

Once the cut set becomes empty the current segment is completed, and the segmentation process continues from the first found new segment base. Once all new bases have been iteratively processed, the segmentation is completed, resulting in a collection of segments that are the branches of the tree, and their topological connections. The segmentation procedure and bifurcation detection is visualized in **Animation IV**.

2.3.2 Updated segmentation

Although the original segmentation procedure worked well the topology it resulted in, was often unnecessarily complex and in many cases the maximum branch order was excessively high. *I.e.*, what the human eye would consider a single branch/segment, was often presented by several consecutive segments, each increasing the branch order. Furthermore, the use of fixed size surface patches was inefficient as the radius parameter had to be set in accordance to the diameter of the smallest branches to be reconstructed, resulting in an excessive number of patches on the larger diameter tree parts, such as the stem. While an update was designed, a key goal was to make TreeQSM more robust in terms of the input parameters given by the user, one of which was the originally fixed surface patch radius.

In an updated version of the reconstruction method, first presented in (Calders et al., 2015b), segmentation is carried-out twice, enabling the use of variable-sized surface patches on the latter run. On the first run a fixed — typically larger — radius is used for the spherical environments, and the segmentation will result in a crude approximation of the branching structure. The size of the user-selected surface patch radius for the first segmentation is expected to be larger than, in most cases, with the original segmentation procedure, thus expected to make the overall segmentation result less sensitive to the selected value. In addition, the initial large radius neighbourhoods allow larger gaps in the point cloud data to be traversed, connecting otherwise separated components. Should separate components remain, they are connected to the closest component by modifying the neighbourhood relation. The resulting segmentation of the tree is further refined, such that each segment extends to the furthest tip of any of its children, lowering the number of individual segments and the maximum branch order, thus overall simplifying the topology.

Next, the segments' shapes are analysed and a non-fixed value for the surface patch radius is determined for each point cloud element, so that the radius decreases linearly when moving upwards in the tree, along the branch, or to a larger branching order. However, all radius values have to be in a user-defined interval, by default derived from the parameters PatchDiam2Min and PatchDiam2Max. As a result of this procedure, each point cloud element has a radius value attached to it that is suitable to expressing the level of detail in the respective part of the tree.

The second segmentation is very similar to the original segmentation process, but now the radius of the spherical environments varies according to the value stored in the point cloud elements selected as center points. Segmentation corrections similar to the first run, are applied to the result of the second run, again lowering the maximum branch order. Furthermore, in order to prevent too large cylinder radius values, child segments are extended with the neighbour relation back into the parent segment, and the extension is excluded from both segments.

2.4 Cylinder reconstruction

Each segment is iteratively reconstructed as a collection of consecutive cylinders — or any other geometric block as shown in **Article III**. The process begins from the base of the segment, once again by selecting a certain number of layers of neighbouring surface patches to form the current sub-segment,[2,3] which is a set of surface patches reconstructed as a single cylinder. Thus the number of layers in the sub-segment determines the approximate length of the resulting cylinder. One of the inputs of the reconstruction procedure is the ratio between the radius and length of a sub-segment, determining the required number of layers in a sub-segment.

The geometry of a sub-segment is used to derive initial values for the iterative least-squares (LS) cylinder fitting (Lukács et al., 1998). The mean point of the complete sub-segment is used as an estimate for a point on the cylinder axis. Then the sub-segment is divided into top and bottom parts using the surface patch layers, and the respective mean points of the parts are connected to estimate the axis direction. Mean distance from the surface patch center points to the line defined by the point and direction estimates, is used as a radius estimate. The final fit is solved iteratively using the Gauss–Newton method. The length of the final cylinder is received by projecting the sub-segment points onto the axis, and computing the difference between the extreme values.

After a segment has been completed, it is possible to do segment-level corrections using simple heuristics, such as limiting the radius ratio, or difference in axis direction between consecutive cylinders. Such corrections are sometimes necessary because of errors in the segmentation or gaps and noise in the point cloud data.

[2] Also called a *subregion* in (Calders et al., 2015b).

[3] In practice, the surface patch sets forming the layers are stored during the segmentation process, and they are utilized when creating sub-segments.

As presented in (Calders et al., 2015b), the current version of the reconstruction procedure performs the following segment-level corrections for branch segments:

1. The radius of the first cylinder of a segment is lowered to match the radius of its cylinder parent, if it is exceeded.
2. Radii smaller than a parameter r_{min} will be fixed to r_{min}. The default value of the parameter $r_{\text{min}} = 2.5$ mm.
3. Finally, a parabola is fitted to the relative-position-along-the-segment – radius data:
   a) Only the first three quarters of length are used, as the values in the last quarter are usually more noisy.
   b) An additional data point is fixed at relative distance 1 with a radius r_{min}.
   c) The resulting parabola is scaled with two factors to receive local lower and upper limits for allowed radius values.
   d) Values outside the bounds are moved to the closest boundary.

For the stem segment the procedure is otherwise the same, but rather that fitting a parabola, a piecewise linear fit is used instead. The curve points for the fitting are computed as averages from a certain number of consecutive cylinder radii values.

Once all segments have been reconstructed, it is possible to perform tree-level corrections as well, such as analysing if there is too much space between consecutive cylinders inside the branches, or between child and parent cylinders. Should such gaps exist, it is possible to try to fill them by extending the existing cylinders in their axis directions, or by introducing additional cylinders whose parameters are derived from the cylinders on opposite sides of the gap. **Animation VII**/**VIII** shows the full process of fitting cylinders, segment-level corrections and filling gaps, using generated point cloud data and a real reconstructed cylinder model.

2.4.1 Alternative reconstruction

Although, the previous section only considered circle-based cylinders, the presented reconstruction approach also allows the use of other geometric primitives, or even segment-level reconstruction such as triangulation. Circular cylinder, elliptic cylinder, polygon-based cylinder, circular cone fitting and segment-level triangulation were tested and compared against one-another in **Article III**. The triangulation approach is similar to the one used to reconstruct complex shapes of tree stumps in (Liski et al., 2013).

In **Article III** the included geometric primitive fitting methods were implemented as iterative, thus relying in initial values. The sensitivity of each method was tested using by simulating error in the initial values during reconstruction. Furthermore, testing was performed with eight generated stem models and eight scanned English Oak tree stems. The effect of lowering data quality on the methods was also tested, by lowering the simulated scanning resolution and by introducing gaps in the point cloud data. The article mainly focused on tree stems, but a reconstruction test was carried out with a single generated full tree model.

2.5 Reconstruction procedure and result evaluation

As mentioned above in addition to the point cloud data, the reconstruction procedure also has some additional input parameters. In the current version[4] of the TreeQSM method additional input parameters are the following:

PatchDiam1 Minimum distance between any two surface patch center points in the first segmentation step. The parameter is a fixed value as patch size is constant in the first segmentation step. By default the radius of the spherical environments, used to form the surface patches in the first segmentation, is derived from the PatchDiam1 value by adding 2 cm.

PatchDiam2Min Minimum distance between two surface patch center points in the second segmentation step.

PatchDiam2Max Maximum distance between two surface patch center points in the second segmentation step. More specifically the minimum value is set to branch end points and the maximum value to the stem base. Values for the rest of the parts are interpolated depending on branch order and radius. By default the radius of the spherical environments, used to form the surface patches in the second segmentation, is derived from the PatchDiam2Max value by adding 1 cm.

lcyl Approximate ratio between the length and radius of all the produced cylinders.

FilRad Relative factor for excluding outlier points in sub-segments during the cylinder fitting procedure. If the distance between a point and the center line-segment of a sub-segment is larger than the estimated sub-segment radius, the point is then excluded.

As a result of the TreeQSM process the point cloud data is reconstructed as a QSM, that is a collection of cylinders with geometric and topological information. Compared to the input point cloud data with disconnected 3D points, the data dimensionality and semantic meaning has increased significantly. In a QSM, the position, orientation and size of each cylinder, together with branch indices and orders and reference to the parent cylinder are stored in the model. This comprehensive description of the structure of a tree can be used to derive properties, such as, branching angles, branch lengths, tree height, volume and taper curves for both the stem and any other branch. **Animation V**/**VI** shows the transformation of a real forest plot with several oak trees into a collection of QSMs, and showcases some of the information that can be further derived from those models.

[4] TreeQSM version 2.30

Once a tree model has been successfully reconstructed, goodness of fit estimation should be performed, the same way as in any other mathematical modelling application. In the realm of QSMs goodness of fit estimates can be used to compare several models reconstructed with differing input parameter values, and also to analyse the variation associated with the stochastic nature of the selection of the surface patch seed points. The latter means that even with the same input parameters and point cloud data, the repeated reconstructions are likely to vary both in terms of topology and geometry. Goodness of fit estimates can be used, *e.g.*, to select a single QSM from a pool of candidate reconstructions, when necessary.

Although it is not strictly a goodness of fit measure, the variation associated with multiple reconstructed models with the same input parameters, can be used to assess the robustness of the reconstruction and a level of certainty for the reconstruction. If the variation is low – *e.g.*, in terms of the total volume of the QSM, the number of individual branches, or the distribution of volume in different branch orders – the segmentation and surface reconstruction procedures have likely been carried out robustly and uncertainty is low. In such a case, selecting a single model for further computations or storing, is simple as all the models are expected to be similar. On the other hand if the variation is high, the selection of the input parameter values should be re-evaluated, to see if higher robustness and level of certainty is achievable.

Visual inspection of the point cloud data compared to the QSM is feasible with smaller trees, but becomes more difficult and time-consuming with increased complexity. The current implementation of TreeQSM uses the average distance from point cloud elements to the closest cylinder, to estimate goodness of fit. The single value is easy to compute and to use to compare different reconstructions. However, additional metrics for assessing reconstruction results should be studied and tested.

2.6 Validation of the TreeQSM method

Quantitative structure modelling seems to be a giant leap in the field of forest measuring and tree modelling, as it offers access to properties previously unavailable, extremely laborious or slow to measure, and properties that can be hard to measure with suitable accuracy. Comprehensive tree models are readily available as long as one can acquire a point cloud of the tree. However, before these claims can be verified, it is necessary to validate the accuracy of the models resulting from QSM reconstruction.

In most applications validation can be done with either real field data, or with simulated data. The level-of-detail incorporated in the simulations determines, how well the computational results are likely to correspond to field results. A laser scanner is essentially a ray tracer: given a direction a scanner determines how far a beam can travel in that direction. Computationally tracing a beam through a scene with complex geometry can be slow. Thus, as a coarse approximation of laser scanning, it is possible to generate random points on the surface of the geometry in the virtual scene. Such an approach would be faster than ray tracing simulated laser beams, but also does not consider the occlusion caused be geometry and the positioning of the scanner in the scene. With both the ray tracing and random sampling approaches it is possible to add simulated measurement noise to measurements. However, only in the case of the former is it possible to limit the noise to the direction of the beam.

Early validation of the reconstruction method was carried out in **Article I**, using a simple manually-constructed cylinder model. Rather than simulating TLS, random points were generated on the surfaces of the cylinders, with a certain level of simulated noise. The generated point cloud was reconstructed and the resulting cylinder model was compared to the original one. The results showed that volume difference was 7.4 %, while the total cylinder length was off by only 1.0 %. However, in retrospect the point cloud generation was extremely unrealistic as it didn't account for self-occlusion and measurement geometry, resulting in even measurement coverage on all parts of the simple test model.

In order to move towards more realistic simulated point cloud data a simple TLS simulator, based on ray-tracing was written in MATLAB and utilized for the first time in **Article III**. The simulator allowed exact measurement geometries be defined, and multiple scans be combined, similarly to the real procedure, while the scanned object consisted of an arbitrary number of triangles. Although occlusion effects and measurement geometry was accounted for in the study, effects like beam divergence, registration errors and environmental factors, such as wind and fog were still not considered.

As shortly mentioned in Sect. 2.4.1, the main goal of **Article III** was to compare the accuracy performance of several geometric primitives when reconstructing tree stems, from either simulated or scanned point cloud data. The main hypothesis of the publication — and what turned out to be true — was that the circle-based cylinder would be the most stable of the primitives, in terms of required initial value accuracy, as well as, data quality and angular point coverage. This was especially evident in the test with a complete tree model, where the simplest shapes (circular and elliptic cylinders) resulted in the lowest difference in reconstructed volume, demonstrating that more complex primitives and reconstruction methods require a fuller measurement coverage than what is usually achievable with branches above the scanner height.

While considering only stems, the difference in volume and surface area were minimal between the reconstruction approaches with both the real stems and the generated stems, deliberately designed to contain exaggerated features. Thus, it is safe to say that if the stem does not contain, *e.g.*, bulges or buttress roots the circular cylinder is a sufficient choice as a geometric primitive in terms of accuracy, and a great choice due to the level of stability and data requirements. That being said, naturally there are applications that focus on cross-section shape, and for these applications the polygon-based cylinder and segment-level triangulation offer the best level-of-detail. As discussed in **Article III**, models combining several primitives or reconstruction approaches are also viable, for example trees with buttress roots.

In terms of volumetric accuracy the QSM reconstruction approach was comprehensively first tested with real TLS data by Burt et al. (2013), using TLS data from three different vegetation types. The trees were isolated manually from the multi-position scans and reconstructed. TLS simulation was carried out on the reconstructed scene, and the trees were reconstructed a second time. The volume of the first and second order reconstructions were compared, and the results showed an average difference of 10 %. The authors also state that the most of the error is caused by point cloud registration errors rather than the QSM reconstruction.

In (Smith et al., 2014) the QSM reconstruction method was modified to reconstruct 13 extracted root systems as a combination of triangulated meshes and cylinders. Reference volumes were measured by displacement, and on average the root system volume was underestimated in the model by 4.4 %. Moreover, the authors also discuss that the use of QSM reconstruction also provides more detailed and important information, *e.g.*, surface are, branching angles and fork count.

Kaasalainen et al. (2014) studied how accurately change in tree biomass can be detected with the help of QSM reconstruction of multiple co-registered point clouds of the same objects, with changes occurring in between. The study consists of two case studies: 1) a single aspen branch scanned four times in a laboratory, with cuts made between scans; and 2) a field study with a single maple tree scanned five times over a two-and-a-half year time span. In the first case study branch volume and length estimates were compared to manual reference measurements (and an alternative volume estimation method, *triangulated irregular network*). The results showed that change in both volume and length is detectable from QSMs, the former with up to a 12 % error, while the latter had a larger error, 8–27 %. The high level of error in branch length change was suggested to be caused by short branches (length $<$ 5 cm), poorly visible in the TLS data, but that the error would be smaller with complete trees as such branches would contribute less to the total volume. The second case study, showed that numerical values can be obtained for occurred change also for complete trees, and that large shed branches can be identified from the QSMs.

AGB estimated from reconstructed QSMs was compared to manual biomass measurements gained through destructive sampling and estimates computed with allometric models by Calders et al. (2015b). Furthermore, manual DBH and tree height measurements were compared to values computed from QSMs. The study contained a total of 65 trees from three different Eucalypt species. The results showed that the measured and estimated DBH values had a coefficient of determination $R^2 = 0.97$ and a RMSE of 2.39 cm, while the same values for tree height were 0.98 and 0.55, respectively. For the biomass comparison the coefficient of variation of the root-mean-square error (CV(RMSE)), and the concordance correlation coefficient (CCC), were used. The former had a value of 16.1 % and the latter a value of 0.98, showing that all the attributes could be accurately estimated from QSM reconstruction made from TLS data. The QSM based estimates were especially more accurate than allometric equations that were also compared against the reference measurements. Where QSMs overestimated biomass by roughly 10 %, allometric equations underestimated the quantity by 30–40 %, especially with larger trees.

The same dataset and biomass estimation results together with an additional dataset with three new species were part of another study the following year, Hackenberg et al. (2015a), where the QSM reconstruction approach presented in this thesis was compared to another reconstruction procedure called *SimpleTree*. The reference biomass of the 36 trees of the new dataset was also measured through destructive harvesting (Hackenberg et al., 2015b). The CCC values for the three new species were *E. fordii*, 0.80; *P. massoniana*, 0.99; and *Q. petraea*, 0.57. For the first two the results were good, but for the third species the volume was overestimated on average by 20 %, although better results might have been possible with further parameter tuning.

Raumonen et al. (2015) presented the preprocessing step for extracting individual trees, but also contained some biomass-based validation, including with a set of 15 English oak trees (*Quercus robur* L.). Rather than destructively measuring reference biomass, allometric equations developed specifically to that forest plot and that species, were compared to QSM derived estimates. TreeQSM input parameters were not optimized and the same values were used for all the trees. The results showed that on a tree-level biomass was either over- or underestimated by around 25 %. On the forest plot-level biomass was overestimated between 15 % and 19 %.

First validation with destructively harvested tropical tree species was performed recently by Gonzalez de Tanago Menaca et al. (2017), with a dataset of 20 trees from three different forest regions. Allometric models were also tested against the QSM reconstruction approach, and the results showed that QSM-based reconstruction outperforms even the most accurate allometric model, with respective CCC values at 0.95 and 0.89. Also noteworthy, is the fact that TLS was performed in very dense forests under leaf-on conditions, and that the tree were extremely large, with DBH values larger than 70 cm. As TLS measurement density was quite low at the top of such tall trees, and poor reconstruction results are to be expected under such conditions, branches with diameters smaller than 10 cm were discarded from both the destructive measurements and the QSM-based volume estimates. Instead their contribution to the volume was estimated using an expansion factor.

All the above research goes to show that QSMs can be used to accurately estimate, *e.g.*, total branch length, tree volume and AGB. Although the estimates are not always accurate down to a single percent, they are often far better than the existing alternatives – allometric models in terms of accuracy, and destructive sampling in terms of nature preservation, time consumption and cost. Furthermore, these are only a fraction of the tree properties that can be estimated from reconstructed QSMs.

3 Species recognition with TreeQSM

QSMs give access to numerous tree properties, and one application that can benefit greatly from that, is tree species recognition. Similarly, as a biologist would determine the species of a tree in the forest, based on certain key factors, species can be determined from attributes derived from reconstructed tree models. In their current form, QSMs do not contain all the information that a biologist might rely on — *e.g.* bark colour and texture, or leaf shape. However, QSMs offer access to other geometric and topological attributes that can even be invisible to the human eye, but still allowing species recognition.

Feature spaces — an abstract concept vital for QSM-based species recognition — is presented in Sect. 3.1. **Article IV** describes an example of automatic, massive-scale QSM-based species classification, while the results are extended and discussed in Sect. 3.2

3.1 Feature spaces

TLS data is in its core three-dimensional data, if you ignore the intensity data. Each point cloud element is simply a range measurement in a given direction. Thus, there is no way of knowing whether any two elements hit the same object, whether a tree, branch or a leaf, even if the directions of the measurement beams differ only by a little. The segmentation process of the TreeQSM method aims to reconstruct the connections between the individual point cloud elements, first identifying the tree and then further separate the stem and branches. These added connections for the point cloud elements can be viewed as an increase in the data dimensionality.

The cylinder reconstruction process, on the other hand, reduces the quantity of data required for describing the tree (million TLS points *versus* thousand cylinders). At the same time the cylinders provide an approximation of the surface of the branches, and the recorded tree topology, *i.e.* the branching structure, is also transferred to the cylinder structure. The resulting data structure, *i.e.* the QSM, is a compact and high-dimensional representation of the tree, suitable for computing countless tree features, previously unavailable or extremely laborious to measure, especially from standing trees. QSMs give access to geometric properties, *e.g.* tree volume, surface area, branch length, branching angles, and taper curves for stem and branches, and topological properties, *e.g.* branch order distribution, maximum branch order and the number of child branches for each branch.

The computed features map the tree model into a *feature space*, allowing for example the comparison of several models in that space. The features can be common tree attributes, such as tree heigh, or quite complex and abstract, like the ratio of the total branch length and the average branching angle of the first-order branches. Furthermore, feature spaces can be multidimensional simply by mapping multiple attribute values of the same tree to different dimensions. Examples of one-dimensional features spaces, and how three selected QSMs map to those spaces, are shown in Fig. 3.1. The included features correspond to the features to be listed in Table 3.1. The key in the figure is that in some feature

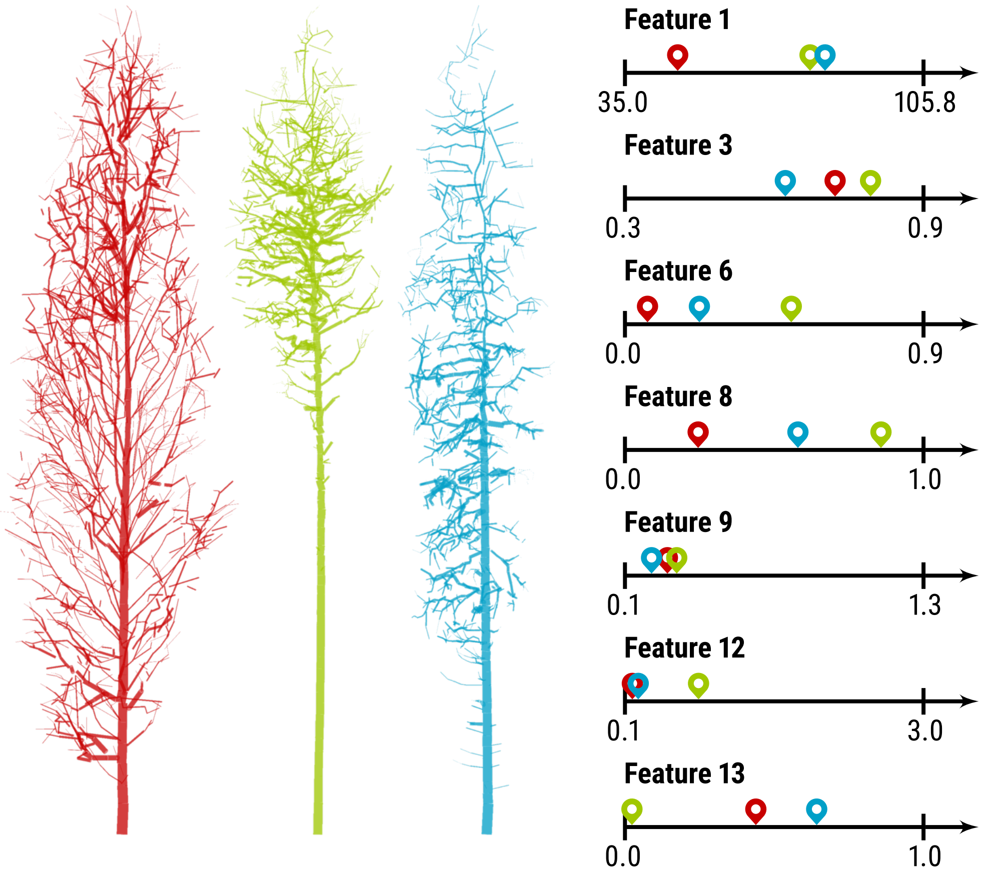


Figure 3.1: Examples of feature spaces for three reconstructed QSMs. From left to right, the selected QSMs are the same Silver birch, Scots pine and Norway spruce models featured in Fig. 4 of **Article IV**. The models are colour-coded to match the pins on the feature space axis.

spaces these three models are clearly separated, while in others some or all of them are very similar. Furthermore, the order in which they map to the space varies from feature to feature.

Feature spaces – called *space of observables* in the publication – of reconstructed tree models were applied to produce stochastic structure models (SSMs) in (Potapov et al., 2016). SSMs are structure models similar to QSMs, but rather than reconstructing an existing tree, SSMs are created simulating tree growth, using a model such as the LIGNUM growth model (Perttunen et al., 1996). In the LIGNUM approach, a tree model consists of parts that grow and can bifurcate to create new parts, at every time step. Originally, the growth control parameters were deterministic, however in the SSM approach some of the growth parameters are stochastic and optimized by minimizing the distance between distributions computed from the generated tree and a reference QSM, *i.e.* by measuring distance in selected feature spaces. As a result the generated tree model is stochastically similar to the reference QSM, but not a direct clone in terms of geometry and topology.

Later, in (Potapov et al., 2017), the method was further developed as a generator for morphological clones of tree models. The procedure, called *Bayes Forest Toolbox*, is written in MATLAB and is available for download.[1] The toolbox allows the user to generate any number of stochastic clones of a reference QSM.

3.2 Classifying tree species

As more TLS data and reconstructed QSMs became available from a variety of tree species, the natural question became, how much variation there was on the newly available tree properties, both intra-species and inter-species. The hypothesis in **Article IV** was that the latter would be considerably higher, allowing the use of QSM-derived properties as classification features in species recognition applications.

A massive dataset provided by the Finnish Natural Resources Institute (LUKE), consisting of over 1200 trees and 3 tree species — Silver birch (*Betula pendula* Roth), Scots pine (*Pinus sylvestris* L.) and Norway spruce (*Picea abies* [L.] Karsten) — from 3 single-species and 2 mixed-species forest plots was used in the study. 15 classification features were chosen based on existing research and trial-and-error to maximize species separation in the corresponding multidimensional feature space. The included classification features are listed in Table 3.1 and presented visually in **Animation XI**. Further information can be found in **Article IV**.

During the research process also the properties listed in Table 3.2 were considered, but excluded for the reasons given in the table. The most common cause for exclusion was the lack of separation between species. Fig. 3.2 visualizes the level of variation in the excluded features per species, similarly as Fig. 3 in **Article IV** for the included species.

Three different training-based classification methods were tested and compared in the study. The k-nearest neighbour, multinomial regression and support vector machine (SVM) based methods were included. Furthermore, three different kernel functions were tested for the SVM method. In comparison to one-another, the methods gave consistent results, indicating that trees of the included species vary significantly on the selected feature spaces.

The included classification features were designed to be scale-independent, and most of them are also unitless. The scale-independence was achieved by normalization by, *e.g.* the DBH or the tree height. The ultimate goal was that regardless of tree age — and thus the absolute size — all trees of the same species would result in similar feature values. Fig. 3.3 shows the distribution of two feature values with the trees of each species separated into three height intervals.

[1] Bayes Forest Toolbox: `https://github.com/inuritdino/BayesForest`

Table 3.1: List of included classification features. For feature with units, the unit is given in brackets at the end of the description.

ID	Name	Description
1	Stem branch angle	Median of the branching angles of the 1st order branches in degrees. 0 is upwards and 180 downwards. [°]
2	Stem branch cluster size	Average number of 1st order branches inside a 40 cm height interval for 1st order branches. Each branch can only belong to one interval.
3	Stem branch radius	Mean ratio between the 10 largest 1st order branches measured at the base and the stem radius at respective height.
4	Stem branch length	Average length of 1st order branches normalized by DBH.
5	Stem branch distance	Average distance between 1st order branches computed using a moving average with a window width 1 m. If window is empty average distance in window is set as half of window width.
6	Crown start height	Height of first stem branch in tree crown relative to tree height.
7	Crown height	Vertical distance between the highest and lowest crown cylinder relative to tree height.
8	Crown evenness	Crown cylinders divided into 8 angular bins. Ratio between extreme minimum heights in bins.
9	Crown diameter / height	Ratio between crown diameter and height.
10	DBH / height ratio	Ratio between DBH and total tree height.
11	DBH / tree volume	Ratio between DBH and total tree volume. [m^{-2}]
12	DBH / minimum tree radius	Ratio between DBH and the minimum of the vertical bin radius estimates.
13	Volume below 55 % of height	Relative cylinder volume below 55 % of tree height.
14	Cylinder length / tree volume	Ratio between total length of all cylinders and total tree volume. [m^{-2}]
15	Shedding ratio	The number of branches without children divided by the number of all branches in the bottom third.

Table 3.2: List of excluded classification features. The last column **ER** lists the *exclusion reason* that is either the index of an included feature that correlated too highly, or NS for *no clear separation*.

ID	Name	Description	ER
16	Volume above 0.65 of height	Relative cylinder volume above 65 % of tree height.	13
17	1st and 2nd order volume ratio	Ratio between the volumes of 1st and 2nd order branches.	NS
18	DBH / branch length	Ratio between DBH and total length of all branches.	NS
19	20 % volume from bottom	Height from bottom at which 20 % of branch volume is reached.	13
20	20 % volume from top	Height from top at which 20 % of branch volume is reached.	13
21	Branch / stem volume	Ratio between branch and stem volume.	NS
22	Mean branch radius / DBH	Ratio between the average branch cylinder radius and DBH.	NS
23	Top and middle third radii ratio	Ratio between the radii estimates of the top and middle vertical bins.	NS
24	DBH / maximum tree radius	Ratio between DBH and the maximum of the vertical bin radius estimates.	NS
25	Density of middle third	Density of a vertical bin is computed as the ratio between the cylinders inside a bin and the cylinder defining the bin.	NS
26	Middle / top third density	Ratio between the densities of the middle and top vertical bins.	NS
27	Stem branches / tree height	Ratio between the number of stem branches and tree height.	NS
28	Stem branch median distance	Median of the pairwise stem branch vertical distances normalized by tree height.	NS
29	Min / max angular volume	Ratio between the minimum and maximum volume of 8 angular bins around the stem.	NS
30	Angular volume std	Standard deviation of the angular bin volumes.	NS
31	Min / max angular branch length	Ratio between the minimum and maximum branch length of the angular bins.	NS
32	Angular branch length std	Standard deviation of the angular bin branch lengths.	NS

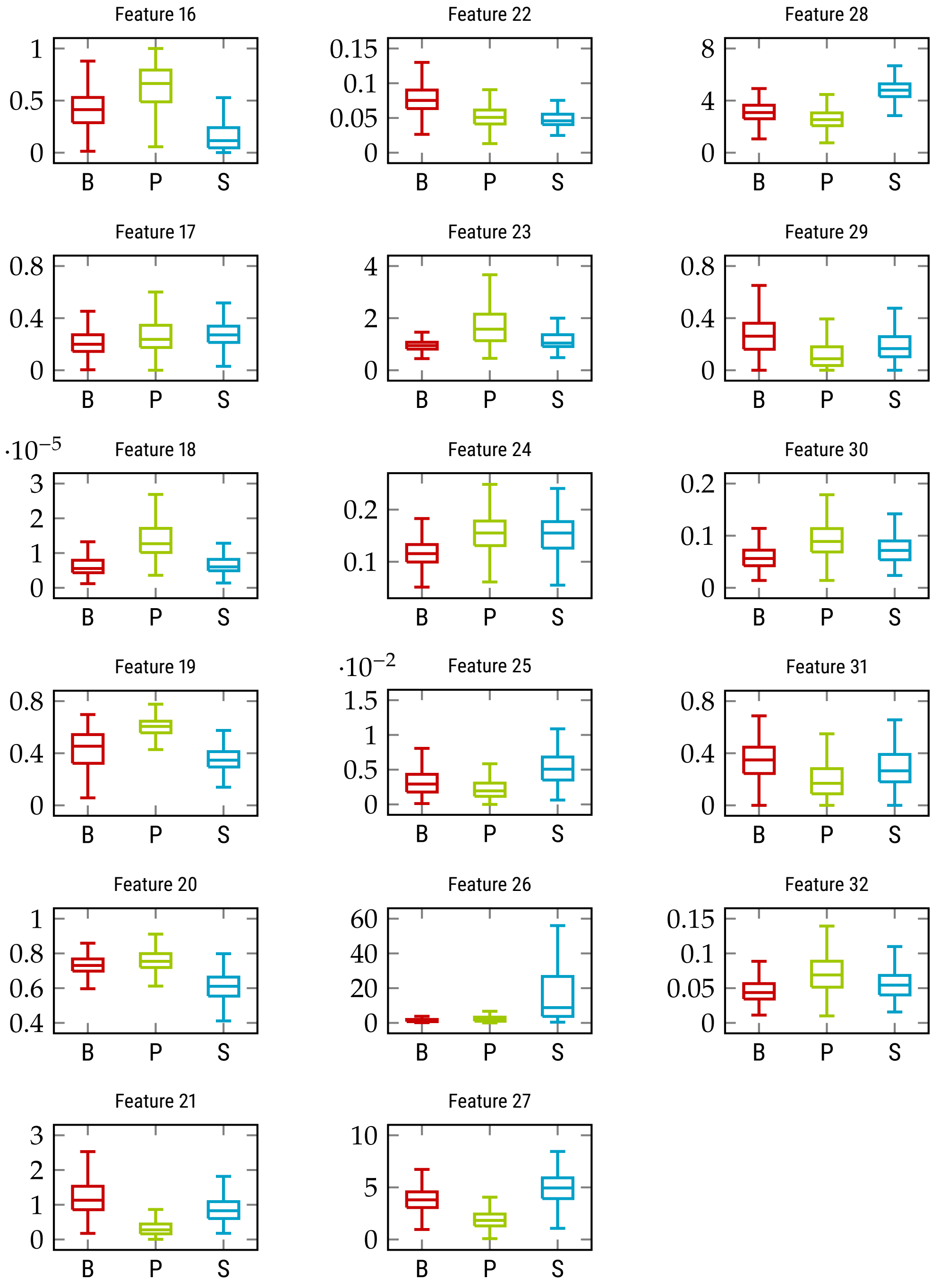


Figure 3.2: Variation in feature values of the excluded features. The vertical line inside the box is the median. Box limits give the 1st and 3rd quartiles of the distribution and the whiskers extend to 1.5 times the distance between the 1st and 3rd quartiles, or the distribution extremes.

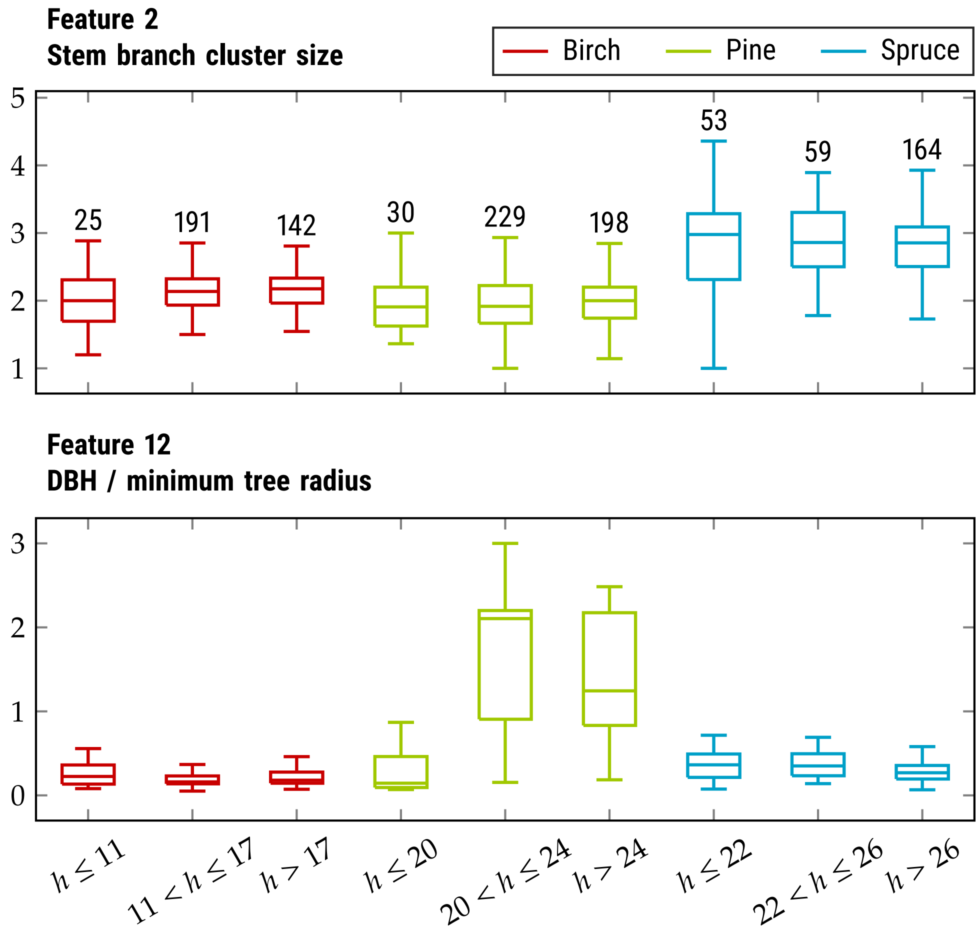


Figure 3.3: Two feature value distribution by tree height intervals. The vertical line inside the box is the median. Box limits give the 1st and 3rd quartiles of the distribution and the whiskers extend to 1.5 times the distance between the 1st and 3rd quartiles, or the distribution extremes. The number of samples in each height bin is listed above the whiskers on the top set of axis.

Table 3.3: Pairwise correlation of the included classification features in percentages. The values are also colour-coded, so that, zero percentage is pure green and 100 percentages is pure red.

	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	
1	40	51	58	40	15	17	2	16	48	31	9	41	45	12	**1**
2		41	37	33	42	42	19	17	31	35	34	38	10	24	**2**
3			62	56	16	17	5	9	52	42	7	28	51	14	**3**
4				68	11	10	20	12	61	47	12	13	71	32	**4**
5					3	3	24	0	48	81	12	0	77	19	**5**
6						100	42	55	23	13	67	67	24	69	**6**
7							43	53	25	12	68	68	24	69	**7**
8								10	11	28	27	31	18	30	**8**
9									21	16	29	24	19	39	**9**
10										22	14	35	44	6	**10**
11											2	1	60	6	**11**
12												61	31	72	**12**
13													5	49	**13**
14														46	**14**

The intervals were chosen, for each species separately, so that each bin would have a sufficient number of samples. For Feature 2 the scale-independence works well, as the variation in feature values in each of the height bins for each species is lower than between the species. Although, there is a little more variation in the bin of the smallest spruces ($h \leq 22$) than the other two spruce bins. An opposite example happens with Feature 12, where the value distribution for the smallest pine trees differs quite a lot from those of the larger pine bins. Furthermore, the distribution of the small pines is quite close to those of the other two species, making it improbable to correctly identify the species of small pines using Feature 12.

In mathematical classification applications it is important that all the selected features remain pairwise uncorrelated. With certain classification methods including additional highly correlating features does not affect the accuracy in any way, however with other methods, correlation may result in unwanted bias in the relevant importance of features, and the classification accuracy. Fig. 3.3 lists the correlation values for each of the included classification feature pairs. The correlation coefficients are expressed in terms of magnitude, without noting whether the linear correlation is positive or negative.

There are two feature pairs that have a high correlation level. Features 6 and 7 correlate perfectly with this dataset. Feature 7 is the height of the crown normalized by the total tree height, and Feature 6 is the similarly normalized starting height, defined by the lowest point of the branch cylinders. Thus, the values of the two features always sum up to one, resulting in perfect negative correlation, unless the lowest branches grow or bend downwards relative to their connection point in the stem. The branches of all the included three species grow upwards, resulting in the perfect correlation. The two features were still both included in the study, as the correlation is expected to be lower if additional species would be included.

In retrospect, the definition of Feature 6 could be changed as the lowest normalized height of any branch tip. The updated definition would result in the same value as the old one in cases where the lowest branches grow downwards. With upwards growing branches the value would also correlate less with the value of Feature 7. How much this would affect the overall classification accuracy, remains untested.

The other feature pair with a high correlation coefficient, 81 %, is the pair (5,11),[2] where Feature 5 is the average distance between stem branches, and Feature 11 is the ratio between the tree DBH and total tree volume. The correlation between these two feature values for the included species remains unexplained. Interestingly, the correlation coefficient for Feature 5 and Feature 14 is also moderately high, 77 %, suggesting that there is a connection between the average distance of stem branches and the ratio of total branch length and total tree volume.

Further studies with a wider array of species should be carried out, to see how the correlation values evolve. However as stated in **Article IV**, the similar classification accuracies achieved with feature combinations with and without the highly correlating feature pairs, show that in this case the correlation levels don't have a significant effect on the classification accuracy.

[2] In **Article IV** this pair is reported as (4,11), due to an error by the author.

Modelling leaves

4

One of the ultimate goals of reconstructing and modelling trees, is to understand the ecosystem of trees consisting of interactions with the soil, the atmosphere and the sun. To be able to study and model these interactions both the roots and leaves have to be included, as they are the actuators of most tree-related interactions. However, reconstructing either the roots or leaves especially from standing trees is very challenging. Root systems can be reconstructed, similarly as described for trees in this thesis, if they are first excavated (Liski et al., 2013; Smith et al., 2014). Reconstructing leaves is equally hard but for a different reason as they are visible, but recording their count and absolute positions is extremely laborious.

Terrestrial laser scanning can be used record points on leaf surfaces, but there are several problems: 1) separating returns from woody and leaf material, 2) identifying returns from the same leaf in order to estimate leaf size or orientation, and 3) leaf presence results in a lot of self-occlusion for the tree, and naturally other trees as well, making the accurate reconstruction of the branching structure unlikely from leaf-on scans. Because of the latter, one usually performs two sets of scans, one leaf-off to reconstruct the woody structure, and one leaf-on to measure properties of the leaf cover. Many methods exist for measuring the leaf area index (LAI) (Woodgate et al., 2017), but also the distribution of leaf material (Béland et al., 2011; Grau et al., 2017) and leaf orientation (Zheng and Moskal, 2012).

Even a single English oak tree, a species which has quite large leaves, can have between 50 and 120 thousand leaves (**Article V**). As such the exact location of each leaf can be seen as irrelevant, while their underlying distribution is the determining factor in interactions with the atmosphere. Once you have the leaf property distributions, it is straightforward to sample them to generate a leaf cover that follows the distribution, enabling the resulting models to be based on exact leaf geometry as well as more abstract property distributions.

Article V presents an algorithm — Foliage and Needles Naïve Insertion (FaNNI) — for generating non-intersecting leaves that follow user-defined leaf property distributions. Also included in the article is the description of a MATLAB implementation of the FaNNI algorithm (**Software I**). The inputs of the implementation include a QSM, leaf basis geometry, and a total area of leaves to be distributed. In the current version[1] the basis geometry can consist of any number of (3-dimensional) triangles, allowing the use of even very complex leaves. However, the results showed that the basis geometry triangle count heavily affected the computational time required for the leaf generation procedure to run, as intersections have to be detected with all the triangles of all the leaves.

When a single leaf is instanced during leaf generation the basis geometry is scaled, rotated and translated, according to sampled values from user-defined — or default — leaf and twig property distributions. This procedure defines the twig start point, leaf start point, direction, length and normal, shown in Fig. 4.1. Even though a normal is defined for a leaf,

[1] Current release: `https://github.com/InverseTampere/qsm-fanni-matlab/tree/v1.2.0`

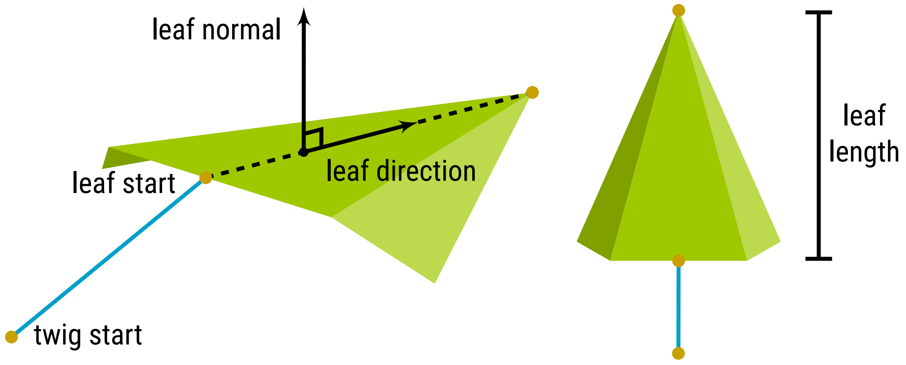


Figure 4.1: Properties of a generated leaf with three triangles in the basis geometry.

the basis geometry does not need to be planar. The normal direction can be viewed as more abstract, fixing a local coordinate system for the leaf together with the leaf direction.

In the implementation the property distributions are defined as MATLAB functions with a fixed interface of input and output arguments, allowing the user to select either the default distribution or substitute one of their own. There are four separate functions for: 1) distributing available leaf area to individual QSM blocks, 2) sampling leaf size (also determines leaf count per block and in total), 3) sampling leaf orientation (direction and normal), and 4) sampling twig parameters (location, orientation, length). The default distribution functions are presented in **Article V**. A common input for functions 1–3 is a *struct* variable containing QSM block properties. Currently the included block properties are fixed in the main file of the leaf insertion implementation, but there are plans to make them user-definable in future versions, in order to make the definition of the distribution functions even more flexible.

As the QSM block properties, such as their relative position on the branch or vertically on the tree, are used to sample values for the required leaf and twig parameters, it is possible to visualize the resulting distribution using the QSM. Examples of such visualizations were shown in **Article V** for the leaf area density distribution (LADD), which describes how the available leaf area will be distributed to the individual blocks. However, it is possible to make similar illustrations for other properties as well. Such properties could include the ones listed above, related to this application, but also unrelated properties such as water content or bark thickness. Fig. 4.2 shows on example of a visualized LADD and a resulting generated leaf cover for the *small oak* in **Article V** – an illustration excluded from the publication for compactness.

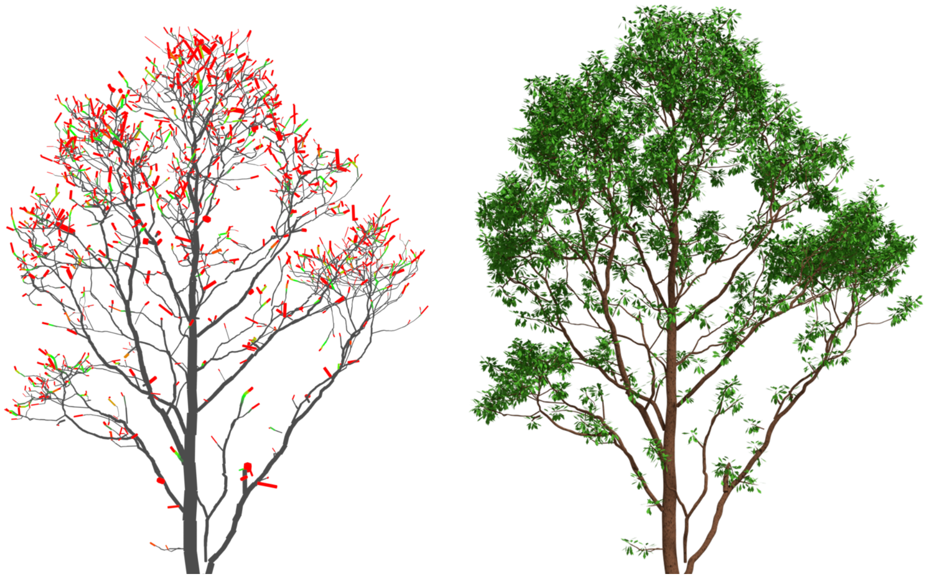


Figure 4.2: An example of a visualized LADD and a resulting leaf cover after leaf sampling for the small oak in **Article V**. The lower parts of the tree have been excluded, and the radii of the cylinders in the LADD illustration have been scaled according to their respective distribution value for a better visualization.

As discussed shortly in **Article V**, one application of the leaf generation algorithm would be to infer the underlying leaf distribution parameters from standing trees. This could be done, *e.g.*, by forming an optimization problem similarly as done in the publication, where parametrized distributions were used to generate leaf covers to explain the manual leaf count and area measurements. Should such an optimization be successful, new understanding could be found on how leaves are distributed along the tree height, and along the branches. Naturally, the distribution would be species and location-dependent, and not universal.

However, should one have the leaf distribution parameters, for a specific species and an area of study, it could also eventually be beneficial to the woody structure reconstruction procedure of those trees, while scanned during leaf-on season. If individual broadleaves could be identified from the point cloud data, and there would be an accurate model of how the leaves of trees of the certain species distribute around the branches, one could infer the location and other geometric properties of the branches holding those leaves. This would allow more of the tree structure to be accurately reconstructed, even with the presence of occluding broadleaves.

5 Dissemination of QSMs

Innovative ways of research result dissemination are becoming increasingly important as the number of publications keeps rising, and on the other hand the availability of suitable dissemination technologies and platforms has risen drastically. As such part of the work done for this thesis consisted of preparing and publishing animations, interactive 3D models and web applications. The results of these efforts are presented in this chapter together with details on how they were produced.

All of the included animations were composed and rendered with Blender,[1] an open-source, cross-platform 3D creation suite for creating 3D graphics and video. Currently Blender ships with two separate render engines, the Internal Render Engine and Cycles, the former of which was used for majority of the animations. Regarding animations, Sect. 5.1 describes how point clouds can be exported and visualized, and Sect. 5.2 describes various options for rendering QSMs in Blender. A list of animations prepared as part of this thesis are listed in Sect. 5.3. Solutions involving interactive 3D models are presented in Sect. 5.4.

5.1 Rendering point clouds for animations

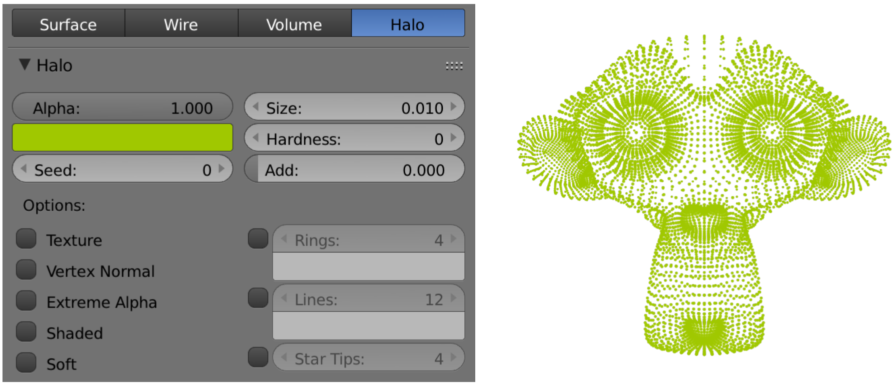
MATLAB was usually used to export point clouds data for animation purposes. The Wavefront OBJ file format was chosen as the migration format between MATLAB and Blender due to its compact representation of vertices. Blender has a built-in importer for OBJ-data. For most of the animations the point cloud data was thinned out by excluding point randomly to achieve more aesthetically pleasing results.

Point clouds can be hard to render as the most common 3D animation software are build for rendering faces and perhaps edges in some cases, but not vertices. There are separate applications meant for visualizing massive point clouds, but problems arrive when trying to mix point clouds with mesh data that consists of, *e.g.*, triangles or Bézier surfaces.

Blender was not designed to render vertices, but there is a workaround that can be used to make vertices visible in a render when using the internal render engine, by using a *halo* material. The material is originally intended to be used to create effects like lens flares, but with the right settings it can be used to make vertices render like diffuse or emission shaded spheres. The settings panel for the Blender halo material is seen in Fig. 5.1.

The *hardness* parameter defines how hard, or distinct, the edges of the spheres appear to be. By setting the hardness to zero the point cloud elements appear solid. Note that if the hardness is set to greater than zero, the object can appear to be out of focus. The *size* parameter of the halo material controls the radius of a single sphere. By default this parameter is relatively large, but if it is set very small the vertices will appear point-like in the render. A fixed value for the parameter is usually not good enough because the appearance of the halo material changes drastically as the camera moves closer to the object; either

[1] Blender 3D, https://www.blender.org/



▼ Halo

Alpha: 1.000

Size: 0.010

Hardness: 0

Seed: 0

Add: 0.000

Options:

☐ Texture

☐ Vertex Normal

☐ Extreme Alpha

☐ Shaded

☐ Soft

☐ Rings: 4

☐ Lines: 12

☐ Star Tips: 4

Figure 5.1: Blender settings for a point cloud material. Left: settings panel for halo material; Right: rendered example point cloud.

the points appear too large when viewed up close, or they won't render at all with large distances. Therefore, the parameter value should be computed as a function of the distance between the point cloud object and the camera. In Blender this can be achieved by *driving* the size parameter with the distance between the camera and the point cloud object.[2]

The following linear relation for the distance $d(f)$ between the object and the camera and the halo material size parameter s in frame f was determined to work well:

$$s(f) = \frac{d_{\max} - d(f)}{d_{\max} - d_{\min}} \cdot (s_{\max} - s_{\min}) + s_{\min}, \tag{5.1}$$

where $d_{\max}$ and $d_{\min}$ are the approximate maximum and minimum distances between the camera and the object, and $s_{\max}$ and $s_{\min}$ the maximum and minimum size parameter values.

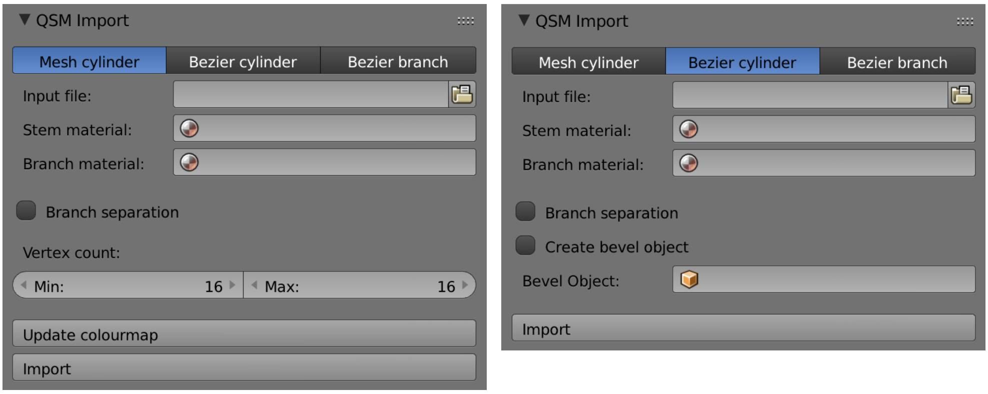
It should be noted that the render size and camera view angle also affect the appearance of the halo material. Thus, the size parameter should only be previewed with final render resolution, and if a scene contains zoom effects the camera view angle should be factored into the driver of the size parameter. However, for many scenes the presented linear relation is sufficient, and it has been used in many of the animations listed in Sect. 5.3.

When rendering animations containing point cloud data it is important to remember that some of the most used video encoders struggle with this type of images. For example the point cloud parts of the 3D Forest Information video suffers from severe decoding artefacts when played on Youtube. Thus, selection of the video encoder and encoding settings are crucial.

5.2 Rendering QSMs

There are a couple of ways to represent cylinders in 3D graphics software such as Blender, *i.e.*, either as mesh objects or as parametrized surfaces received by lofting Bézier curves.

[2] The size could also be key-framed, but a driver updates automatically should changes occur later.



▼ QSM Import

Mesh cylinder | Bezier cylinder | Bezier branch

Input file:

Stem material:

Branch material:

□ Branch separation

Vertex count:

Min: 16 | Max: 16

Update colourmap

Import

▼ QSM Import

Mesh cylinder | Bezier cylinder | Bezier branch

Input file:

Stem material:

Branch material:

□ Branch separation

□ Create bevel object

Bevel Object:

Import

Figure 5.2: User interface of the Blender QSM import add-on. Left: settings for mesh import; Right: settings for lofted Bézier models.

In the early animations the former approach was used, and thus QSM data was exported from MATLAB in Wavefront OBJ format, either with triangular or rectangular faces. During the export procedure it was possible to choose the number of vertices on the top and bottom loops of each cylinder based on the respective radius, thus having less vertices and faces on cylinders with smaller radii. In later videos the cylinders were presented as lofted curves. The QSMs were exported in a custom text file format, where each row described the following properties of a single cylinder, in order with the number of respective parameters in parenthesis: branch index (1), starting point (3), axis direction (3), length (1), radius (1), and optional parameters ($N \in \{0, 1, ...\}$).

A Blender add-on (**Software II**) was developed to automate the QSM import process as both mesh and curve objects. There are three alternatives for importing a QSM: 1) mesh object, 2) cylinder-level Bézier curves, and 3) branch-level Bézier curves. The UI of the add-on is shown in Fig. 5.2. Regardless of the resulting object type, cylinder data is passed to the QSM import add-on in the custom text file format described above. The data file is set in the *Input file* field of the UI. The user can also optionally choose separate materials for the tree stem and branches, assigned automatically during import to the appropriate QSM parts. An option also exist for the user to choose whether the whole QSM is presented as a single Blender object, or should a separate object be created for each branch. Note that the latter is much slower when the number of branches is high.

When importing as a mesh object, the user can choose the minimum and maximum number of vertices on cylinder top and bottom loops. The vertex count is determined by linear interpolation based on respective cylinder radius and the minimum and maximum radius found in the input file, similarly what was done manually in MATLAB.

With the lofting based methods a lofting object (or bevel object in Blender's terminology) has to be selected in the respective field. This object is duplicated at each curve point with possible scaling, and orientated to match the curve tangent at that point. The duplicated instances are then connected to each other, forming the final surface. The bevel object can be any shape, but in all the animations considered here, a closed Bézier circle with the radius at unity was used. Compared to the mesh approach, lofting is more dynamic as in Blender it is possible to change the number of control points on the bevel object, enabling the choice in level-of-detail at any point of the production, and the level could even be made dependent on the distance between the camera and the QSM object, allowing an automated level-of-detail selection.

In the cylinder-level lofted models the axis of each cylinder is presented by a separate Bézier curve with two points, one at the starting point and one at the end point. The curve direction is also set parallel to the cylinder axis direction, making the axis a straight line segment. The radius parameter in both curve points is set to the cylinder radius value, thus resulting in the resulting lofted surface to be a cylinder.

With the branch-level lofted models the same idea is expanded further, presenting a single branch as a lofted Bézier curve, allowing also tapering along the branch. A branch with N cylinders results in a curve with $N+2$ points: $\mathbf{c}_0$ at the starting point of the first cylinder, $\mathbf{c}_i$ at the center point of each cylinder $i \in \{1, \ldots, N\}$, and $\mathbf{c}_{N+1}$ at the ending point of the last cylinder:

$$\mathbf{c}_i = \begin{cases} \mathbf{p}_1 & \text{when } i = 0 \\ \mathbf{p}_i + \frac{1}{2} h_i \mathbf{a}_i & \text{when } 0 < i \leq N \\ \mathbf{p}_N + h_N \mathbf{a}_N & \text{when } i = N+1 \end{cases}, \tag{5.2}$$

where $\mathbf{p}_i$ is the starting point, $\mathbf{a}_i$ the axis direction, and h_i the length of the ith cylinder, $i \in \{1, \ldots, N\}$. Similarly, the left $\mathbf{d}_i^-$ and right $\mathbf{d}_i^+$ curve control points, that determine the curve tangent at the curve points, are computed as follows:

$$\mathbf{d}_i^\pm = \begin{cases} \mathbf{c}_0 \pm \gamma_0^\pm h_1 \mathbf{a}_1 & \text{when } i = 0 \\ \mathbf{c}_i \pm \gamma_i^\pm h_i \mathbf{a}_i & \text{when } 0 < i \leq N \\ \mathbf{c}_{N+1} \pm \gamma_{N+1}^\pm h_N \mathbf{a}_N & \text{when } i = N+1 \end{cases}, \tag{5.3}$$

where $\gamma_i^\pm$ is a parameter determining how far from the curve points the control points are placed, determining the steepness of the curvature. In the current implementation the following parameter values are used:

$$\gamma_i^+ = \begin{cases} 0.25 & \text{when } i \in \{0, N, N+1\} \\ 0.45 & \text{otherwise} \end{cases}, \tag{5.4}$$

$$\gamma_i^- = \begin{cases} 0.25 & \text{when } i \in \{0, 1, N+1\} \\ 0.45 & \text{otherwise} \end{cases}. \tag{5.5}$$

Given the radius r_i of cylinder i, the radii t_i at the curve points $\mathbf{c}_i$ are set to the following:

$$t_i = \begin{cases} t_0 = r_1 & \text{when } i = 0 \\ t_i = r_i & \text{when } 0 < i \leq N \\ t_{N+1} = \beta r_N & \text{when } i = N+1 \end{cases}, \tag{5.6}$$

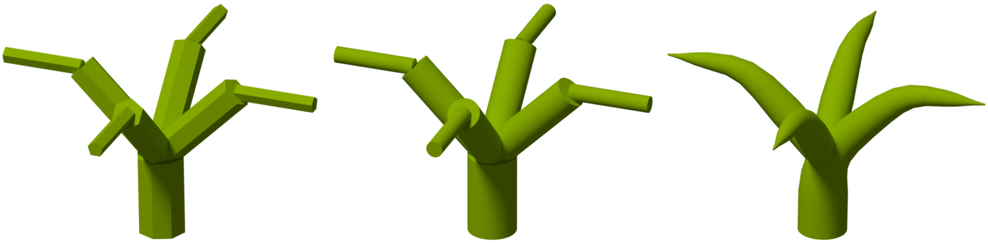


Figure 5.3: Examples of imported models. From left to right: mesh cylinder, cylinder-level, and branch-level lofted models.

where the radius parameter β controls how steeply the branch ends should close. Setting $\beta = 0$ forces the branch end to taper to an apex, whereas $\beta = 1$ results in the radius r_N of the last cylinder. In the current implementation the radius parameter β is fixed at 0.1.

Examples of QSM import results are shown in Fig. 5.3 for all the resulting object types. The vertex number limits for the mesh model were 5 and 8. Because of the low vertex count the mesh model looks less smooth than its lofted counter parts, but with a higher vertex count similar results would be possible. However, with the lofted models you can choose the smoothness level even after the import procedure, for both the branch curve as well as the bevel object.

Currently, it is recommended to use one of the lofted model options over the mesh-based one, as there is a huge difference in computational time. This is because the mesh import option utilizes Blender's built-in computationally intensive operator to create basis cylinders that are transformed to create the final QSM cylinders. The execution time is expected to decrease if the cylinder vertex and face generation would be done manually.

In future versions of the QSM import add-on the selection of the parameter values $\gamma_i^{\pm}$ and β should be exposed to the user, enabling more versatile use. There are also plans to implement the option to import multiple QSMs at once, allowing easy visualizations of forest plots.

5.3 Details of produced animations

Below is a list of animations produced as part of this thesis. Short descriptions of each video are given, together with an overview of how specific parts or effects — such as point clouds or QSMs — were created during the production. To see the animations, refer to the Supporting material section of this thesis.

Animation I - *Creating surface patches*

A short animation on the creation of surface patches and the definition of the neighbour relation. In the animation point cloud elements are represented as individual sphere objects, due to the low point count.

Animation II - *Surface patch characteristics*

The video visualizes how data variation and PCA can be used to determine point cloud or surface patch properties. In the QSM reconstruction algorithm PCA is used to define a dimensionality to surface patches, and a direction to elongated patches, as well as a normal for planar patches. Point cloud elements are visualized as individual spheres.

Animation III - *Locating the trunk and base*

Before QSM reconstruction from point cloud data, lidar returns from ground and surrounding vegetation have to be filtered out. This is done by first identifying surface patches part of the tree trunk by analysing patch geometry with PCA. Ground exclusion is based on the neighbour relation, as it enables moving along the underlying surface, similarly to a propagating wave, while preventing trunk parts being labelled as ground. Once the trunk and ground points are classified, branches are defined to be the remaining unclassified points.

In the animation the halo material approach from Sect. 5.1 was used to visualize real, thinned-out point cloud data of a pine tree. The propagating colour wave was achieved by utilizing Blender's dynamic paint tools and invisible *brush* objects.

Animation IV - *Automatic segmentation*

The animation begins with a point cloud of a tree with a known set of surface patches that form the base of the trunk, and visualizes how the neighbour relation can once again be used to select *layers* of surface patches and move along the trunk. Bifurcation detection is presented through examples.

The tree model in the animation was generated with the Blender Sapling add-on.[3] The point cloud was generated in Blender by using a particle system, that was then instanced to receive a mesh with vertices. The point cloud elements were then generated with another

[3] Sapling: https://wiki.blender.org/index.php/Extensions:2.6/Py/Scripts/Curve/Sapling_Tree

hair particle system with the particle count matching the vertex count, and a sphere as the *dupli object*. The colour changes relied on several dynamic paint systems.

Animation V/VI - *3D Forest Information*

A demonstration of a tree extraction procedure, the QSM reconstruction approach and its applications, using real point cloud data from an oak tree point cloud. In the video the thinned-out and ground-excluded input point cloud is introduced and shown to transform into cylindrical QSMs. Computed values such as branch count, volume, and stem taper curve for individual QSMs are shown at the end.

Point cloud data was rendered using the halo material approach. The QSMs were exported from MATLAB in Wavefront OBJ format, as the import add-on was yet to be created. Thus, the low vertex count of the mesh cylinders is evident in parts of the animation.

Animation VII/VIII - *Cylinder reconstruction*

The video describes in detail the QSM reconstruction process steps between the segmented point cloud and the resulting cylinder model. Each segment is divided into sub-segments of certain length, each of which is then reconstructed by fitting a single cylinder in the LS sense. After a segment has been completed, possible gaps in the model can be removed by introducing additional cylinders, or by modifying existing cylinder parameters. Furthermore, heuristics can be used to ensure too abrupt changes in the consecutive radii in the cylinders of the segment.

A tree model was generated with Blender's Sapling add-on, and simulated laser scanning was performed on it in MATLAB. The resulting point cloud was reconstructed as a QSM. The point cloud and cylinder data were exported into Blender. The halo material approach was used to visualize the point cloud.

Animation IX - *Visualizing reconstructed tree models*

The animation begins with a demonstration of a tree getting laser scanned from various directions, showing how the individual scans can be co-registered. Cylinder-level and branch-level lofted QSMs are featured in the video, together with leaves generated with an early version of the FaNNI leaf insertion algorithm (**Article V**). Both the QSM and the leaves are visualized with various textures and compared to each other.

The point cloud was exported from MATLAB where the vertices were first ordered according to angle in polar coordinates with respect to the virtual scanner. This allowed the point cloud to appear as a wave, by using a *build modifier* in Blender. The points were visualized using the halo material approach. The animation was created at the same time as the first version of the QSM import add-on and all the QSMs are lofted Bézier models.

Animation X - *Quantitative Structure Models*

This compilation video mostly features graphics from previous videos, as a summary of the details of the QSM reconstruction approach, applications that are enabled by QSMs, and visualization capabilities. Original parts of the video feature statistics derived from reconstructed QSMs from the Punkaharju forest plots in **Article IV**, and demonstrate how different tree species are separable for accurate species classification.

The data points in the statistics section were exported from MATLAB, and imported as mesh vertices and respective *shape keys*. This allowed the smooth animation between different feature dimensions with simple controls.

Animation XI - *Tree species recognition with quantitative structure models*

The video was prepared supplementary to **Article IV**, and it visualizes how the 15 classification features are computed, with the help of an example QSM. Later in the video classification results are visualized for each feature separately, using additional example QSMs, one of each species.

Data for the box plots was exported from MATLAB and imported as shape keys of a mesh object. *Empty objects* were then *vertex parented* into the mesh vertices, and these object were *hooked* to the appropriate box plot geometry. This complex approach allowed smooth transitions between consecutive features simply by animating the shape key values.

5.4 Interactive 3D models and example tree gallery

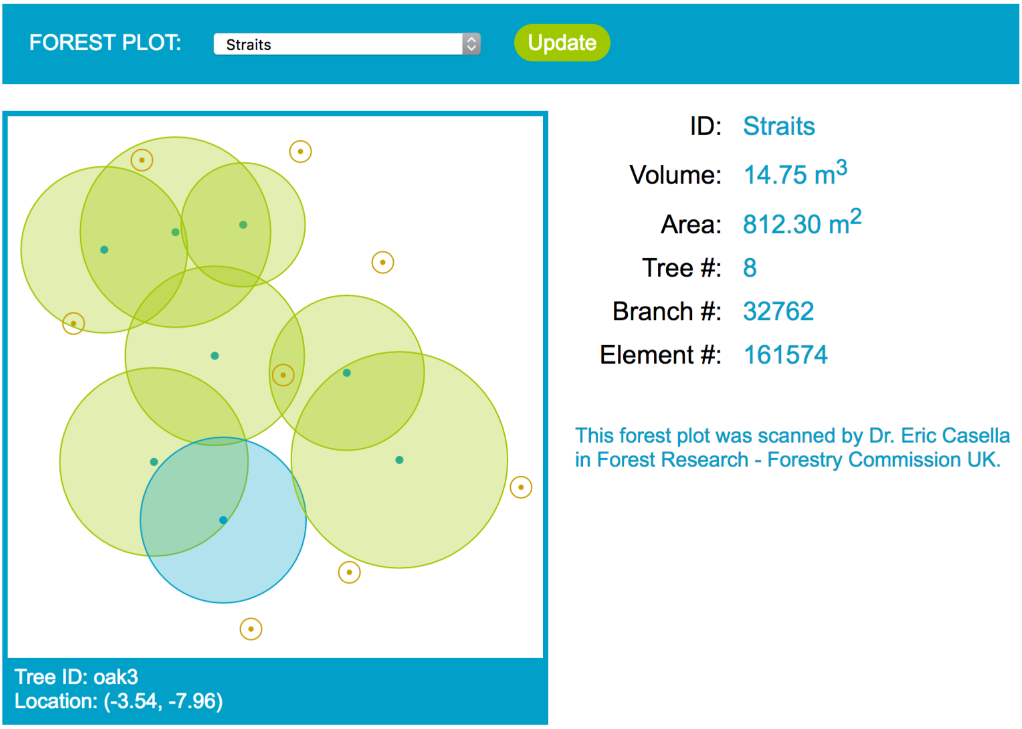
Besides videos, QSMs have been disseminated as interactive 3D models. These models are powered by web graphics library (WebGL), which is a JavaScript-based technology that allows 3D graphics to be displayed directly in a compatible web browser, without the user having to install any add-ons. Although, it would be possible to write a custom viewer utilizing the WebGL application programming interface (API), for simplicity an existing web service called SketchFab[4] was used. A 3D model can be uploaded to the service in various formats, such as, Wavefront OBJ. After that, displaying the model is only a matter of getting the appropriate embedding code to a web page. The 3D model player offered by SketchFab includes controls to rotate, pan and zoom the scene.

A more feature-rich web application was also produced as part of this thesis, designed for visualizing forest plot maps and individual trees and their properties. The application is titled *Tree gallery*.[5] The page consists of three sections: 1) forest plot selector and tree map, 2) interactive tree model player and basic tree properties, and 3) branch/cylinder size distribution bar plot.

In the first section, shown in Fig. 5.4, the user can select a forest plot by name in a dropdown menu. After the selection a tree map of the plot will be drawn below, together with the scanning locations. Basic properties, such as, tree branch and cylinder count, are

[4] SketchFab, `http://www.sketchfab.com`

[5] Tree gallery demonstration, `http://math.tut.fi/inversegroup/treegallery`



FOREST PLOT: Straits

ID:	Straits
Volume:	14.75 m^3
Area:	812.30 m^2
Tree #:	8
Branch #:	32762
Element #:	161574

This forest plot was scanned by Dr. Eric Casella in Forest Research - Forestry Commission UK.

Tree ID: oak3
Location: (-3.54, -7.96)

Figure 5.4: Forest plot section of the Tree gallery demonstration.

listed next to the map. In the tree map the center point of each tree is marked as a blue dot, with a larger green circle around it, showing the estimated radius of the tree. In this example the radius was estimated as the maximum horizontal distance between a cylinder and the base of the tree stem. The tree map is interactive and can be used to select a tree to be inspected more closely in the other two sections.

The tree section has similar controls for selecting a single tree from the active forest plot. Once a tree is selected basic properties including area, volume and branch count, are printed in the section. Furthermore, the respective interactive tree model is loaded from SketchFab. The size distribution section is also updated to show the distribution values of the selected tree.

In the last section the user can select between length, area and volume distributions, and whether the distributions are given as a function of branch order or cylinder radius. There is also a choice between absolute and relative units. Both the tree map plot and the size distribution bar plots are implemented using the Data-Driven Documents (D3.js) JavaScript library.[6]

As the application was used only as a demonstration, data from only one forest plot are available. The forest plot and the tree models are the same used in the 3D Forest Informa-

[6] Data-driven documents — D3.js, https://d3js.org/

tion video, described in Sect. 5.3. With larger forest plots it would be possible to pan and zoom the tree map, and in future versions the map could be integrated to an open map API, allowing the use of satellite images or terrain maps.

Discussion 6

At its core, TreeQSM was designed to be a method for reconstructing tree architecture in the form of structure models, in a non-destructive manner. However, over time the goals have evolved to some day have models that also describe tree dynamics and functions, such as growth, nutrient fluxes and interaction with the soil and atmosphere. In fact, steps have already been taken towards these goals, with TreeQSM used to study tree growth (Kaasalainen et al., 2014) and QSMs being combined with growth models (Potapov et al., 2017). **Article V** presented a method for generating a leaf cover for a QSM — an advancement towards atmospheric interactions — but also showcased how non-structural information can be attached to QSM primitives, to describe various distributions. Such an approach could be instrumental at some point, when including nutrient flows into the models.

The accuracy and speed of TLS instruments are increasing while the prices keep decreasing, making them more readily available to forest scientists, the forest industry and people inventorying forests. At the same time QSMs, produced by TreeQSM or other methods, are getting better in terms of accuracy. Destructive sampling is still, and might forever remain, more accurate but at the same time extremely time consuming and not viable for larger scales. One could use allometric models in larger scales, but studies have shown that the accuracy of reconstructed tree models already far exceeds the accuracy of allometric models (Calders et al., 2015b). As the benefits are clear, it is expected that QSMs will be the near-future-way of measuring forests.

QSMs have allowed the quantitative analysis at tree-level, and in the future they can do the same at forest-plot-level and forest-level. Previously, when one wanted to validate a theory related to tree structure, the most time-consuming process was the gathering of validation data from dozens or hundreds of trees. The single-species forest-plots described in **Article IV** — with roughly a thousand trees — took only two days to scan, and some hours to reconstruct as QSMs. Thus, gathering massive amounts of validation data can become extremely fast with these technologies. Furthermore, the resulting models are comprehensive and compact to store, therefore allowing the same measurements to provide various validation metrics for multiple studies, without the need to necessarily measure again. The format for sharing structural and functional tree information should be standardized as soon as possible, to help researches share and utilize an increasing number of reconstructed tree models.

Three aspects of the likely future applications of QSMs are discussed next in separate sections. How the availability of comprehensive tree models is likely to change how forest inventories are performed, is described in Sect. 6.1. Relating to the need for a standard way to present tree-related information, database solutions are discussed in Sect. 6.2. The section also describes an example database structure and a related software demonstration. Finally, visualization related applications are reviewed in Sect. 6.3.

6.1 Towards comprehensive and fully automated forest inventories

The original TreeQSM procedure in **Article II** was designed to work on a single tree point cloud, and to reconstruct a single tree. Although, the initial version was able to automatically detect and exclude lidar returns from the ground and undergrowth, results from point clouds with multiple trees would have had unexpected results. At that time, the research questions were more focused on what information could be extracted even from a single tree. However, as the reconstruction procedure began to produce promising results, it became relevant to study whether the reconstruction could be made to work on a forest-plot level.

Raumonen et al. (2015) presented a pre-processing step for the QSM reconstruction, that segmented a forest-plot-level point cloud into subsets of individual trees. The same concepts of surface patches, patch geometric characteristics and neighbour relation were re-purposed to locate stem bases in the forest-plot-level data, and eventually to exclude ground and undergrowth returns and isolate individual trees. After that it is simply a matter of running the reconstruction procedure on each tree-level point cloud separately, effectively as a whole, moving from the forest-plot-level point cloud data into a collection of QSMs forming the forest plot.

In **Article IV** an additional step was studied, where based on features computed from QSMs — that were automatically separated and reconstructed — were used to identify tree species. The initial results were promising, and showed that species-specific information is stored in QSMs, in a form that is easily accessible. From an application perspective, combining the proposed tree extraction, QSM reconstruction and species reconstruction procedures, allows the computation of the species distribution and numerous tree attributes or quality estimates aggregated per species, just from a forest plot point cloud. Albeit, certain requirements have to be met, such as certain point density and preferably leaf-off conditions, we are moving closer to extremely automated forest inventories, where the main limitation seems to be the area of forest that can be scanned in a given time frame. At the same time, the actual definition of taking forest inventory is changing; The previously recorded few key values per tree are just part of what is included in the detailed snapshot of the forest structure, when using QSM reconstruction. It might even be the end for characterizing trees with the traditional, easy-to-measure terms, such as, DBH and tree height, as QSMs might be able to offer properties that are similarly compact but even more descriptive.

How much forest area can be covered with a suitable resolution, for example in a single day, depends on the forest density and the present tree species as the level of branching density can be estimated from those. As part of the field measurements for **Article IV** the Silver birch and Scots pine forest plots were both scanned during a single day, still with some time to spare as the scanning of the Norway spruce forest plot also started the same day. Combined, the first two forest plots required 20 scanning locations and included more than 800 trees.

It is easy to see that, if not this particular QSM reconstruction approach then some other TLS-based methodology will be the future of forest inventories. As the results of the power of what can be reconstructed and estimated from TLS data spreads wider, there should be no reason to perform manual DBH and tree height estimates over TLS scans. Especially, when you compare the number of trees that are visible even from a single scan, and how much time it will take to sample the same trees manually. At some point the scanning process can become even faster, if the limitations with drone technology – available flight time and the carrying capacity – are removed, and if any form of mobile scanning can match the accuracy of static TLS measurements.

In the near future of forest scanning, there might be a rapid move from the currently most common single-wavelength instruments to their hyper-spectral counterparts. Many hyper-spectral scanners have already been developed (Danson et al., 2014; Hakala et al., 2012), and the progress is sure to continue, with instruments getting cheaper and smaller, with increases in both the spatial and spectral accuracy.

Hyper-spectral lidar measurements could be used to augment the recorded forest snapshots with chemical information about different trees, branches and leaves. Allowing for example easier and more accurate separation of leaves and woody structure, estimating leaf distribution parameters (see Sect. 4), and mapping water content and chlorophyll levels. Such information could in turn be used to assess the quality of the wood and the growing conditions, and modelling and reconstructing the functional processes of the tree.

6.2 QSM database

QSM data are well-suited for relational databases, and an initial demonstration was carried out as part of this thesis. QSMs contain cylinder, branch and tree-level details, which can be stored in separate tables to efficiently avoid redundant information. Fig. 6.1 shows the database schema used in the QSM database demonstration, together with some additional tables that were planned but not implemented at this stage.

The database was built on the assumption that a tree has a fixed species and a location (at least on any given time). Multiple QSMs can be reconstructed of any tree with various reconstruction methods, using multiple laser scans, or scan sets, as input data. Furthermore, a QSM consists of a number of branches, which in turn consist of cylinders in the demonstration, while other shapes are also possible (**Article III**).

The database demonstration was web-based and built upon an structured query language (SQL) database and an Apache server, with the back-end written in PHP: hypertext preprocessor (PHP). The front-end was a combination of hypertext markup language (HTML) and JavaScript. The UI consisted of two sections: 1) choosing and filtering input data, and 2) selecting output computations performed on the selected data.

QSMs and their parts could be filtered according to various conditions, such as, forest plot name, tree name, species, height and volume. Furthermore, branch-level filterable properties included branch length, order and volume, and cylinder-level radius, length and volume. For simplicity and ease-of-use any number of filters could be added in arbitrary

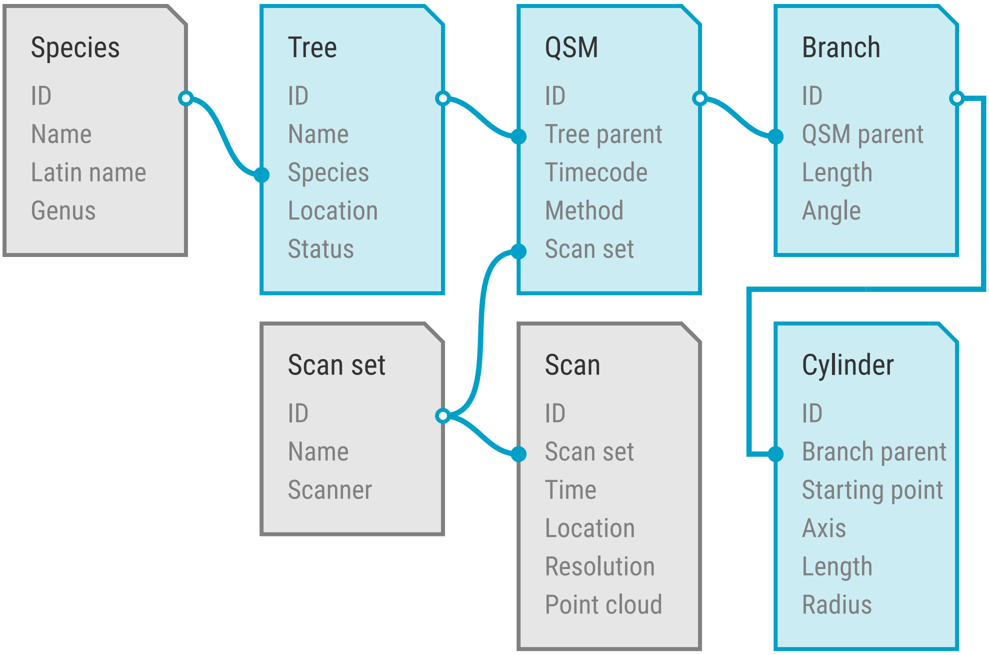


Figure 6.1: Database table and property diagram. Tables included in the database demonstration are shown in blue and planned tables are shown in grey. Foreign key references are shown as lines.

order, but the UI did not allow the filters to be combined by other operators than the logical `AND` operator. There are however plans to allow the use of more advanced operators and a way to determine their processing order.

The possible output computations included basic properties, species distribution and size distributions. The first was a panel containing numerical values of the included models, such as tree, branch and cylinder count, volume and area. Species distribution listed tree counts per species. The functionality and customizability of the size distribution output was similar to the one presented for the Tree gallery in Sect. 5.4. *E.g.*, the user is able to select between branch order and cylinder radius based distributions, and length, area and volume distributions. The resulting values could be displayed either as a table or a bar plot.

The ambitious goal of the database demonstration was that everything was accessible through the website. However, that meant that computations had to be performed on the server-side, or that the data had to be exported in text format. The problem with the former is that all of the output computation offered by the system must be known and implemented beforehand, by the system administrator which would be tedious and not very dynamic, as more and more applications for the models arise. With the latter, there were questions of exportation format and convenience, as the data would have to be further transferred from the web browser to some data processing software. The compromised usability of possibly numerous query–export repeats would certainly reduce the user base.

One solution for further development would be to apply interface-based design to the database, making it accessible thought a web interface, having data exportation in a format like JavaScript object notation (JSON) the only option. Add-ons could then be created for data processing software, allowing the user to make queries and access the return data directly, *e.g.*, from the command prompt of MATLAB.

If necessary the interface could also be utilized by a website to offer basic computations, like checking the number of available models from a given area, and visualize both tree maps and individual tree models, similarly to the Tree gallery in Sect. 5.4. With the help of a custom WebGL viewer parts of the selected QSM could be hidden or coloured, for example, to separate different branch orders. Interaction is also possible, allowing the user to select branches or blocks in the 3D viewer, and displaying properties of the selection.

The applications for the database could include simple data mining to better understand how genetics and environmental factors affect tree growth and interaction with their surroundings. For example, a database would allow an easy comparison of how different forest management approaches affect trees in the area, that are genetically close to each other. A growing database of QSMs could also serve as training data for automated species recognition during TLS-based forest inventories, with the methods presented in **Article IV**. Furthermore, with the help of data mining better species classification features could be discovered from a database, making the species recognition even more accurate.

Potapov et al. (2017) presented a method to generate morphological tree clones using statistics extracted from reconstructed QSMs. Generated tree models that are not exact copies of one-another are used, *e.g.*, in biological simulations and visualization applications such as video games. Connecting such a generator on the proposed database would allow the user not only to download reconstructed trees, but also easily generate any number of tree models that are statistically similar to a model pool selected from the database.

6.3 Visualization and virtual reality

TreeQSM was developed to accurately and quantitatively reconstruct the geometry and topology of a tree. As such, it was not required for the resulting surface to be continuous or, *e.g.*, to realistically represent the texture of the tree bark. In fact, regardless of data quality a tree model consisting of any geometric primitives, can never be continuous. When the approach was first introduced, many potential appliers of the method criticised the method for such reasons, for producing unrealistic-looking results. Branches in the tree models consisted of several separate cylinders, rather than eye-pleasing continuous surfaces. On top of that, gaps in the models were common, as the gap filling and other refinement procedures were yet to be introduced. However, the discontinuous nature of the models was by-design, as the goal at that point was to only reconstruct parts that are well covered in the input point cloud data, differentiating from methods such as (Côté et al., 2009) where single returns, or *attractors* could generate a branch.

With later versions of the reconstruction approach gap filling and tree- and segment-level heuristics help make the models more continuous, but poorly-covered parts are still excluded from reconstruction, as the uncertainty of any reconstruction attempt would be too large. This commitment to relying only on the input data, is one of the factors explaining why the reconstructed volume is usually underestimated in studies (*e.g.* Burt et al., 2013).

The cylinder-based models were well suited for the initial research questions as they are fast and simple to do computations with. However, as the variety of applications utilizing QSMs has grown vastly, so has the demand for more realistic visualizations. Therefore, as presented in Sect. 5.2 some visualization approaches were adapted for presenting QSMs in more visually pleasing ways. As shown in **Animation IX** individual cylinder parameters can be used to parametrize Bézier surfaces, presenting branches as continuous surfaces. Furthermore, both the cylinder surfaces and Bézier surfaces can be textured, *e.g.*, with a bark texture, and textured leaves can be included using **Software I**. Some minor gaps can still remain in the model between parent and child segments, but overall the results are much more pleasing to the eye.

Realistic visualization can be used for example in gaming and virtual reality (VR) applications (Raumonen et al., 2016). For the former, reconstructed tree models can reduce time required for asset creation and the level of realism on nature modelling, as artist are not required to create individual tree models by hand. A suitable number of tree models could simply be downloaded from a database, should one exist. Furthermore for VR, tree and forest reconstruction allows the virtualization of complete scenes, either to market them or to conserve a 3D snapshot of the real location. Tourists could study locations — in detail — they are considering visiting to help with the decision, or revisit sites they have already travelled to and re-experience the scenery.

7 Conclusion

One of the goals of this thesis was to study what tree attributes can be estimated from TLS data, in a non-destructive manner. As a result a novel approach to reconstruct comprehensive 3D reconstructions of trees from terrestrial laser scanning point clouds data has been developed. The TreeQSM software has been implemented in Matlab and it has been shared open-source for anyone to use. The TreeQSM reconstruction automatically segments the tree-level point cloud into branches, using a few tuning parameters chosen by the user, and outputs a model that contains the geometry and topology of the tree present in the data. As the reconstruction is comprehensive, meaning that all the trees sufficiently covered by measurements are reconstructed and included in the model, numerous tree-, branch- and sub-branch-level attributes can be estimated from the model. Tree attributes from single-valued properties, such as tree volume or above-ground biomass, to more complex properties like branch size and branching angle distributions, are all accessible in the produced models.

At the core of the QSM framework are the notions of quantity and comprehensiveness; Reconstruct all branches well covered by the TLS range measurements, to produce a single model describing the full geometry and topology. Previous reconstruction approaches were designed to get estimates of single tree properties such as tree count in a given area, DBH, tree height or stem taper. With TreeQSM the goal is to reconstruct a complete semantically-rich representation of the tree, which allows the computation of not just one specific tree property but most of them. The process is data-driven, meaning that only parts with enough measurements are reconstructed. Although in newer versions of TreeQSM small gaps can be filled using interpolation, the basic design principle is to ignore poorly covered parts and to not guess and try to explain all the isolated lidar returns.

TreeQSM was developed primarily as a tool for forest scientists, foresters and biologists, to answer various questions regarding, *e.g.*, tree structure, growth and interaction with their surroundings. In terms of usage, the TreeQSM software package has become widely-spread and well accepted into operational use, especially by forest scientists. Even before the source-code was released, the software was used by dozens of collaborators all over the world, with more certainly to follow with the software freely available for download and modification. So far TreeQSM has been successfully used to estimate at least branch size distributions (Raumonen et al., 2011), DBH, tree height, tree, trunk and branch volumes and respective biomasses. More advanced applications include the study of scaling laws (Sarmiento et al., 2015), detecting growth in trees over time (Kaasalainen et al., 2014), effects of forest management (Juchheim et al., 2017), generating morphological tree model clones (Potapov et al., 2017), and the estimation of key climate variables (Calders et al., 2015a). Additionally, promising research with the help of QSMs is actively conducted on phenotyping based on structure, crown development mechanics, optimizing the lidar measurement geometry to minimize occlusion, optimizing destructive sampling, studying wind mechanics of trees and on calibrating airborne and space-borne measurements by upscaling terrestrial results. The number of tree properties and applications continues to increase as more professionals adopt the use of TreeQSM, and as measurement technologies (such as HSL) allow the inclusion of additional functional or structural data.

A QSM consists of a collection of geometric primitives – most often circular cylinders – with known geometric parameters and a parent−child primitive relations. As this is an approximation of the real tree surface, the model-based error, *i.e.*, the error related to the selection of the geometric primitive was studied in **Article III**, in order to see the differences between various geometric primitives and to validate the selection of the circular cylinder as the default geometric primitive. The results showed that the circular cylinder is an excellent choice as the geometric primitive as the volume of the reconstructed stem models was overestimated only by 1.36 % on average, and the DBH was underestimated by only 1.12 %. This combined with the stability and the low data quality requirements of the circular cylinder, make it far superior with respect to the other tested primitives – elliptic and polygon-based cylinders, cones and segment-level surface triangulation.

Extensive validation of the reconstruction approach has been carried out in various studies presented in Sect. 2.6, using both simulations and field measurements. Many of the studies suggest that the expected volume error is ≤ 10 % for trees (Burt et al., 2013) and only 4.4 % for root systems (Smith et al., 2014). Above-ground biomass can be estimated non-destructively and more accurately than with the current industry standard – allometric models. The CCC values of TreeQSM-derived estimates and estimates received by destructive harvesting can be as high as between 0.95 and 0.98 (Calders et al., 2015b; Gonzalez de Tanago Menaca et al., 2017). However, for some species, such as the *Q. petraea*, parameter tuning might be needed as CCC values can be as low as 0.57 (Hackenberg et al., 2015a).

Even forest-plot level point clouds with multiple trees can be reconstructed without user interaction, by combining the presented reconstruction approach with a pre-processing step that automatically identifies and extracts individual trees (Raumonen et al., 2015). In traditional forest-plot surveys, performed manually by an expert, an important aspect is the generation of the plot-level species distribution. This thesis aimed to show that identifying tree species is possible using reconstructed QSMs. **Article IV** showed that it is possible to find multiple tree properties that are species-specific, allowing the species of a tree to be recognized automatically based on them. Given suitable training data, various classification methods could be used to classify tree species with an accuracy up to 96.9 %, in a study with over 1200 trees. QSM reconstruction and automatic species recognition could enable large-scale automated forest inventories in the future. Creating and maintaining an open database of reconstructed tree models would be a clear benefit for forestry, forest sciences, and in particular species recognition, as an increasing population of training data would allow even more accurate classification.

At least at the moment, the TreeQSM procedure does not reconstruct leaves or needles or estimate their distribution parameters. However, foliage and needles are essential in all the interactions a tree has with the atmosphere, as well as the appearance of the tree. As such foliage is key in both tree-related simulations and visualization applications. In contrast to reconstructing leaves while reconstructing the tree structure, the other option is to generate a leaf cover after the structure reconstruction. **Article V** introduced a procedure, FaNNI, for adding leaves to QSMs, following an arbitrary distribution chosen by user. Leaf-augmented QSMs are a valuable tool, *e.g.*, for radiative transfer simulations, but also a step closer to more realistic visualizations. The FaNNI implementation written in MATLAB is freely available for download. As part of the study it was not possible to find a parametrized LADD to explain measured vertical leaf distribution, and thus further studies are required. If the distribution of leaves could be explained by parametrized distributions, it would be possible to estimate leaf count and area in standing trees with fewer manual samples. Furthermore, the distribution information could be used to make structure reconstruction more accurate, even during leaf-on season.

A key part of this thesis was the visualization and dissemination of the reconstructed tree models and other results. Although QSM reconstruction produces a collection of individual cylinder parameters, it is possible to use the same parameters to parametrize segment-level continuous surfaces, as shown in Sect. 5.2. Even more realistic results can be achieved by texturing the surfaces as shown in **Animation IX**. On the other hand in comparison to realism, QSMs can be used to visualize parameter or resource distributions of a tree as shown in **Article V** and Sect. 4.

The possible applications of reconstructed tree models are clear for forest science and forest management and harvesting. However, businesses in the travel and gaming industries could also utilize the reconstructed models. Environment reconstruction allows the virtualization of for example travel locations, allowing the user to visit the virtual representation either before, after or instead of the trip. For game developers reconstructed tree models can offer speed-ups in the game asset creation process.

The animations part of the thesis feature both point cloud data and QSMs, or their parts, and feature key parts of the reconstruction algorithm and what the results can offer to various fields. Similarly, new web technologies such as WebGL based interactive 3D models (Sect. 5.4) together with interactive distribution visualizations, have been used for dissemination of the results. Currently in science, these alternative ways of result dissemination remain highly underused, compared to traditional static content. Rather than replacing written publications, these technologies should be used in parallel to them, or as supplementary material. The benefits are clear: a wider audience can be reached and mode complex concepts can be presented faster.

7.1 Future work

Comprehensive reconstructed 3D models are slowly but steadily becoming a standard in forest science. Improvements in the reconstruction accuracy and speed are always required, and part of those improvements come directly from advancements in laser scanning technology, as more detailed point clouds translate to more detailed QSMs. However, the main next step is to start focusing on the applications of the models. In their current form structural tree models are sufficient for estimating tree properties, recognizing their species and for simulating radiative transfer and terrestrial, airborne and spaceborne laser scanning.

Although, Raumonen et al. (2015) introduced a preprocessing step to extract individual trees from forest-plot-level point clouds, further development has to be carried out to ensure the approach works on various conditions, despite the various levels of tree density. Improvements are required especially in complex, multilayered, dense forests, where there might be hardly any lidar return from the top parts, or the parts close to the center of the tree crown. Fully accurate, automated tree separation might be impossible, as distinguishing between intertwined branches from two trees is hard even for the human eye, but still there is room for improvement.

In its current form, the TreeQSM reconstruction implementation only utilizes the location information of the input point cloud elements. On top of the intensity values, or the possible hyper-spectral information, there is at least one more easily accessible set of data that could be utilized to improve the reconstruction results: the measurement geometry. Considering the travelling direction the light beam resulting in the range measurement, one could analyse not only what is visible but also what is not and why. For example, if some environment in the 3D scene does not contain any returns, the positioning of the scanners can be used to determine whether that is because no beam entered the environment, or because all of the beams travelled through it uninterrupted.

Measurement geometry analysis could be carried out *in situ*, allowing the scanner operator to view in real-time what parts, *e.g.* voxels, need more coverage and possibly even have positioning recommendations for additional scans. The key would be to visit as much of the space in the scene, and more importantly, from as many directions as possible, to maximize local coverage where possible. During the reconstruction procedure, the most simple form accounting for the beam direction, could mean checking that none of the QSM elements block the path from the point cloud elements to their respective scanners. The next level would be to determine the level of local coverage, and furthermore a level of trust in the corresponding reconstruction. The environments not visited by beams could be marked as *not seen*, and their combined volume could be used to make corrections in, *e.g.*, volume or biomass estimates.

As shortly described in Sect. 2.5 it is key to have methods to assess the quality of QSMs. Studies should be made into what are the best indicators and measures to describe and evaluate the level of success for the QSM reconstructions. Having a goodness-of-fit estimate of the reconstruction, allows the process of reconstructing a single tree, to be repeated and the results compared to each other, to automatically find the best input parameter values and the optimal output. Furthermore, it might be possible to either reduce the number of input parameters for TreeQSM or to simplify the selection of their value, by systematically evaluating the quality of the output QSM they produce.

When the count of trees to be reconstructed is high, or when the reconstruction needs to happen in real-time, the importance of the optimization of computational resources rises. Currently TreeQSM is only implemented in MATLAB, but should the applications require, it would be possible to port it to other programming languages. It would be interesting to see, what kind of speed improvements could be achieved just by using a pre-compiled language.

TreeQSM was not the first attempt to record tree architecture, as detailed in Sect. 1.1, and it certainly was not the last. Novel ways for adding semantics to point cloud data, especially from forests, should be studied. Improvements can always be made in speed and accuracy, but also in comprehensiveness. There is a long way to go before we can record the state of the complex and dynamic system of geometry, topology and chemistry, known as a tree, even on a single time step, yet alone measure the processes inside the surface over time.

That being said, the functional aspect of trees should be integrated into the structure models. Adding leaves to the models was a step in this direction, as leaves interact with the sun and the tree environment. Further adding the flow of nutrients to the model would allow the studying of tree–soil interactions, through simulations. For example, studying how a disruption in the nutrient flow would affect the health of the tree, its appearance and ability to grow leaves.

Full 4D models would also allow the geometric and functional presentations of a tree to change over time, essentially allowing the tree to grow and evolve. In fact, a step in this direction was already taken by our team in Potapov et al. (2017), where growth simulations were used to generate morphological tree clones, using statistics extracted from reconstructed QSMs.

To the author's knowledge, no public database of reconstructed tree models exists at the moment. However, the benefits of such a collection are clear, and thus resources should be allocated for creating such a database. As discussed in Sect. 6.2, the best approach for a database would be to offer an application interface to the data, allowing developers and enthusiasts to create their own applications, rather than trying to built a single user interface for everyone.

Either together with the database development or separately the accuracy of the presented tree species recognition approach, should be tested on a wider array of species. The key limitation for a wider species classification study is the availability of a sufficient number of manually classified QSMs — or point cloud data — of each species, as they are required for the training data.

At the core of the species recognition approach is the concept of feature spaces, that are N-dimensional spaces where a QSM can be mapped to, for an abstract representation. In this particular application the feature spaces were used to separate models reconstructed of different species of trees using various mathematical classifiers. However it would be interesting to study what could be some other uses for feature spaces either in the realm of trees and forest science, or in any other application of information technology. Also interesting is the current procedure of increasing the point cloud data dimension through the reconstruction process, and then decreasing it once more by mapping the tree model to a low dimensional feature space. For certain applications, *e.g.* species recognition, it would be beneficial to know if there exists a short-cut, bypassing the requirement of the reconstruction when there is no other use for the QSM.

TLS combined with QSM reconstruction (and perhaps terrain mapping) allows the virtualization of locations with trees. One logical step to take in the future is to present the reconstructed scene in a VR environment, allowing the user to experience these elements of nature, for example, from viewpoints inaccessible in real life, or to simulate the behaviour of the location in changing conditions. This could mean previewing the effects of landscaping on the health and appearance of the remaining trees.

From a technical point of view, a key question in presenting trees with or without leaves in VR environments, is the level of the required simplifications in order to meet the hardware and software limitations. Trees can have hundreds of thousands, or even up to millions of leaves, and the woody parts require a fair amount of triangles for smooth visualizations as well, making them extremely resource-intensive to render. It is clear that not all of the parts of the tree and all the leaves need to be rendered at full detail, but how to switch between the detailed and the more coarse presentations without disrupting the experience, remains an interesting topic.

Studies have shown that people can experience stress relief and other health benefits simply by walking in a forest. Studies could be made into the health effects of a person visiting a virtual reconstruction of a forest, to see whether the results are similar and if VR could be used as a tool to maintain health.

Further down the road, if technological advancements allow it, could be tree and forest related augmented reality (AR) applications. Either based on pre-scanned locations, or real-time video-based reconstruction performed by the AR equipment. This would allow overlaying information related to a specific tree or the complete forest plot onto the user view. This way the user could identify tree species on the fly, or even see an animation of the surrounding area evolving over time.

Bibliography

Åkerblom, M. 2012. Quantitative tree modeling from laser scanning data. Master's thesis, Tampere University of Technology. http://dspace.cc.tut.fi/dpub/handle/123456789/21012.

Baccini, A., Walker, W., Carvalho, L., Farina, M., Sulla-Menashe, D., and Houghton, R. A. 2017. Tropical forests are a net carbon source based on aboveground measurements of gain and loss. *Science*, 358(6360):230–234.

Beech, E., Rivers, M., Oldfield, S., and Smith, P. P. 2017. Globaltreesearch: The first complete global database of tree species and country distributions. *Journal of Sustainable Forestry*, 36(5):454–489.

Bi, H., Turner, J., and Lambert, M. J. 2004. Additive biomass equations for native eucalypt forest trees of temperate australia. *Trees*, 18(4):467–479.

Bienert, A., Hess, C., Maas, H.-G., and von Oheimb, G. 2014. A voxel-based technique to estimate the volume of trees from terrestrial laser scanner data. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-5:101–106.

Borden, K. A., Thomas, S. C., and Isaac, M. E. 2017. Interspecific variation of tree root architecture in a temperate agroforestry system characterized using ground-penetrating radar. *Plant and Soil*, 410(1):323–334.

Boudon, F., Preuksakarn, C., Ferraro, P., Diener, J., Nacry, P., Nikinmaa, E., and Godin, C. 2014. Quantitative assessment of automatic reconstructions of branching systems obtained from laser scanning. *Annals of Botany*, 114(4):853.

Bucksch, A., Lindenbergh, R., and Menenti, M. 2010. SkelTre - Robust skeleton extraction from imperfect point clouds. *The Visual Computer*, 26(10):1283–1300.

Burt, A. 2017. *New 3D measurements of forest structure*. PhD thesis, University College London.

Burt, A., Disney, M. I., Raumonen, P., Armston, J., Calders, K., and Lewis, P. 2013. Rapid characterisation of forest structure from TLS and 3D modelling. P. 3387–3390 in *2013 IEEE International Geoscience and Remote Sensing Symposium - IGARSS*.

Béland, M., Widlowski, J.-L., Fournier, R. A., Côté, J.-F., and Verstraete, M. M. 2011. Estimating leaf area distribution in savanna trees from terrestrial LiDAR measurements. *Agricultural and Forest Meteorology*, 151(9):1252 – 1266.

Calders, K. 2015. *Terrestrial laser scanning for forest monitoring*. PhD thesis, Wageningen University.

Calders, K., Disney, M., Nightingale, J., Origo, N., Barker, A., Raumonen, P., Lewis, P., Burt, A., Brennan, J., and Fox, N. 2015a. Traceability of essential climate variables through forest stand reconstruction with terrestrial laser scanning. P. 122–124 in *Proceedings of the SilviLaser 2015 conference*.

Calders, K., Newnham, G., Burt, A., Murphy, S., Raumonen, P., Herold, M., Culvenor, D., Avitabile, V., Disney, M., Armston, J., and Kaasalainen, M. 2015b. Non-destructive estimates of above-ground biomass using terrestrial laser scanning. *Methods in Ecology and Evolution*, 6(2):198–208.

Casella, E. and Sinoquet, H. 2007. Botanical determinants of foliage clumping and light interception in two-year-old coppice poplar canopies: assessment from 3-D plant mock-ups. *Annals of Forest Science*, 64(4):395–404.

Chave, J., Réjou-Méchain, M., Búrquez, A., Chidumayo, E., Colgan, M. S., Delitti, W. B., Duque, A., Eid, T., Fearnside, P. M., Goodman, R. C., Henry, M., Martínez-Yrízar, A., Mugasha, W. A., Muller-Landau, H. C., Mencuccini, M., Nelson, B. W., Ngomanda, A., Nogueira, E. M., Ortiz-Malavassi, E., Pélissier, R., Ploton, P., Ryan, C. M., Saldarriaga, J. G., and Vieilledent, G. 2014. Improved allometric models to estimate the aboveground biomass of tropical trees. *Global Change Biology*, 20(10):3177–3190.

Chen, Y., Räikkönen, E., Kaasalainen, S., Suomalainen, J., Hakala, T., Hyyppä, J., and Chen, R. 2010. Two-channel Hyperspectral LiDAR with a Supercontinuum Laser Source. *Sensors*, 10(7):7057–7066.

Cheng, Z.-L., Zhang, X.-P., and Chen, B.-Q. 2007. Simple reconstruction of tree branches from a single range image. *Journal of computer science and technology*, 22(6):846–858.

Côté, J.-F., Fournier, R. A., and Egli, R. 2011. An architectural model of trees to estimate forest structural attributes using terrestrial LiDAR. *Environmental Modelling & Software*, 26(6):761–777.

Côté, J.-F., Widlowski, J.-L., Fournier, R. A., and Verstraete, M. M. 2009. The structural and radiative consistency of three-dimensional tree reconstructions from terrestrial LiDAR. *Remote Sensing of Environment*, 113(5):1067–1081.

Crowther, T. W., Glick, H. B., Covey, K. R., Bettigole, C., Maynard, D. S., Thomas, S. M., Smith, J. R., Hintler, G., Duguid, M. C., Amatulli, G., Tuanmu, M.-N., Jetz, W., Salas, C., Stam, C., Piotto, D., Tavani, R., Green, S., Bruce, G., Williams, S. J., Wiser, S. K., Huber, M. O., Hengeveld, G. M., Nabuurs, G.-J., Tikhonova, E., Borchardt, P., Li, C.-F., Powrie, L. W., Fischer, M., Hemp, A., Homeier, J., Cho, P., Vibrans, A. C., Umunay, P. M., Piao, S. L., Rowe, C. W., Ashton, M. S., Crane, P. R., and Bradford, M. A. 2015. Mapping tree density at a global scale. *Nature*, 525(7568):201–205.

Danson, F. M., Gaulton, R., Armitage, R. P., Disney, M., Gunawan, O., Lewis, P., Pearson, G., and Ramirez, A. F. 2014. Developing a dual-wavelength full-waveform terrestrial laser scanner to characterize forest canopy structure. *Agricultural and Forest Meteorology*, 198:7–14.

Dassot, M., Constant, T., and Fournier, M. 2011. The use of terrestrial lidar technology in forest science: application fields, benefits and challenges. *Annals of forest science*, 68(5):959–974.

Debreczy, Z. and Rácz, I. 1997. El árbol del Tule: the ancient giant of Oaxaca. *Arnoldia: The Magazine of the Arnold Arboretum*, 57(4):3–11.

FAO 2016. *State of the World's Forests 2016. Forests and Agriculture: Land-Use Challenges and Opportunities*. FAO, Rome.

FIBIC, editor 2014. *Value Through Intensive and Efficient Fibre Supply. Programme Report 2010-2013.* FIBIC, Suomi.

Godin, C., Costes, E., and Sinoquet, H. 1999. A method for describing plant architecture which integrates topology and geometry. *Annals of Botany*, 84(3):343–357.

Godin, C. and Sinoquet, H. 2005. Functional–structural plant modelling. *New Phytologist*, 166(3):705–708.

Gomes, J., Velho, L., and Sousa, M. 2012. *Computer Graphics: Theory and Practice*. An A. K. Peters book. Taylor & Francis.

Gonzalez de Tanago Menaca, J., Lau, A., Bartholomeusm, H., Herold, M., Avitabile, V., Raumonen, P., Martius, C., Goodman, R., Disney, M., Manuri, S., Burt, A., and Calders, K. 2017. Estimation of above-ground biomass of large tropical trees with Terrestrial LiDAR. *Methods in Ecology and Evolution*.

Gorte, B. and Pfeifer, N. 2004. Structuring laser-scanned trees using 3D mathematical morphology. *International Archives of Photogrammetry and Remote Sensing*, 35(B5):929–933.

Grau, E., Durrieu, S., Fournier, R., Gastellu-Etchegorry, J.-P., and Yin, T. 2017. Estimation of 3D vegetation density with Terrestrial Laser Scanning data using voxels. A sensitivity analysis of influencing parameters. *Remote Sensing of Environment*, 191:373 – 388.

Hackenberg, J., Morhart, C., Sheppard, J., Spiecker, H., and Disney, M. 2014. Highly accurate tree models derived from terrestrial laser scan data: A method description. *Forests*, 5(5):1069–1105.

Hackenberg, J., Spiecker, H., Calders, K., Disney, M., and Raumonen, P. 2015a. SimpleTree - An efficient open source tool to build tree models from TLS clouds. *Forests*, 6(11):4245–4294.

Hackenberg, J., Wassenberg, M., Spiecker, H., and Sun, D. 2015b. Non destructive method for biomass prediction combining TLS derived tree volume and wood density. *Forests*, 6(4):1274–1300.

Hakala, T., Suomalainen, J., Kaasalainen, S., and Chen, Y. 2012. Full waveform hyperspectral LiDAR for terrestrial laser scanning. *Optics express*, 20(7):7119–7127.

Hopkinson, C., Chasmer, L., Young-Pow, C., and Treitz, P. 2004. Assessing forest metrics with a ground-based scanning LiDAR. *Canadian Journal of Forest Research*, 34(3):573–583.

Hyyppä, J., Hyyppä, H., Leckie, D., Gougeon, F., Yu, X., and Maltamo, M. 2008. Review of methods of small-footprint airborne laser scanning for extracting forest inventory data in boreal forests. *International Journal of Remote Sensing*, 29(5):1339–1366.

Jolliffe, I. 2002. *Principal component analysis*. Springer series in statistics. Springer-Verlag.

Juchheim, J., Annighöfer, P., Ammer, C., Calders, K., Raumonen, P., and Seidel, D. 2017. How management intensity and neighborhood composition affect the structure of beech (Fagus sylvatica L.) trees. *Trees*, 31(5):1723–1735.

Kaasalainen, S., Holopainen, M., Karjalainen, M., Vastaranta, M., Kankare, V., Karila, K., and Osmanoglu, B. 2015. Combining lidar and synthetic aperture radar data to estimate forest biomass: Status and prospects. *Forests*, 6(1):252–270.

Kaasalainen, S., Krooks, A., Liski, J., Raumonen, P., Kaartinen, H., Kaasalainen, M., Puttonen, E., Anttila, K., and Mäkipää, R. 2014. Change detection of tree biomass with terrestrial laser scanning and quantitative structure modelling. *Remote Sensing*, 6(5):3906–3922.

Kankare, V., Joensuu, M., Vauhkonen, J., Holopainen, M., Tanhuanpää, T., Vastaranta, M., Hyyppä, J., Hyyppä, H., Alho, P., Rikala, J., and Sipi, M. 2014. Estimation of the timber quality of scots pine with terrestrial laser scanning. *Forests*, 5(8):1879–1895.

Kelbe, D., Romanczyk, P., van Aardt, J., and Cawse-Nicholson, K. 2013. Reconstruction of 3D tree stem models from low-cost terrestrial laser scanner data. P. 873106–873106–12 , volume 8731.

Kukko, A., Kaartinen, H., Hyyppä, J., and Chen, Y. 2012. Multiplatform mobile laser scanning: Usability and performance. *Sensors*, 12(9):11712–11733.

Lefsky, M. and McHale, M. R. 2008. Volume estimates of trees with complex architecture from terrestrial laser scanning. *Journal of Applied Remote Sensing*, 2(1):023521–023521.

Liang, X., Litkey, P., Hyyppa, J., Kaartinen, H., Vastaranta, M., and Holopainen, M. 2012. Automatic stem mapping using single-scan terrestrial laser scanning. *IEEE Transactions on Geoscience and Remote Sensing*, 50(2):661–670.

Liski, J., Raumonen, P., Kaasalainen, S., Repo, A., Krooks, A., Akujärvi, A., and Kaasalainen, M. 2013. Indirect emissions of bioenergy: detailed analysis of stump-root systems. *Global Change Biology Bioenergy*.

Livny, Y., Yan, F., Olson, M., Chen, B., Zhang, H., and El-Sana, J. 2010. Automatic reconstruction of tree skeletal structures from point clouds. P. 151 in *ACM Transactions on Graphics (TOG)*, volume 29. ACM.

Lukács, G., Martin, R., and Marshall, D. 1998. Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. P. 671–686 in *Computer Vision-ECCV'98*. Springer.

Murray, C. D. 1926. The physiological principle of minimum work applied to the angle of branching of arteries. *The Journal of General Physiology*, 9(6):835–841.

Neubert, B., Franken, T., and Deussen, O. 2007. Approximate image-based tree-modeling using particle flows. P. 88 in *ACM Transactions on Graphics (TOG)*, volume 26. ACM.

Okabe, A., Boots, B., Sugihara, K., and Chiu, S. 2009. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley Series in Probability and Statistics. Wiley.

Olofsson, K., Holmgren, J., and Olsson, H. 2014. Tree stem and height measurements using terrestrial laser scanning and the ransac algorithm. *Remote Sensing*, 6(5):4323–4344.

Perkel, J. 2018. Data visualization tools drive interactivity and reproducibility in online publishing. *Nature*, 554(7690):133–134.

Perttunen, J., Sievänen, R., Nikinmaa, E., Salminen, H., Saarenmaa, H., and Väkevä, J. 1996. LIGNUM: a tree model based on simple structural units. *Annals of botany*, 77(1):87–98.

Pfeifer, N., Gorte, B., and Winterhalder, D. 2004. Automatic reconstruction of single trees from terrestrial laser scanner data. P. 114–119 in *Proceedings of 20th ISPRS Congress*.

Potapov, I., Järvenpää, M., Åkerblom, M., Raumonen, P., and Kaasalainen, M. 2016. Data-based stochastic modeling of tree growth and structure formation. *Silva Fennica*, 50(1).

Potapov, I., Järvenpää, M., Åkerblom, M., Raumonen, P., and Kaasalainen, M. 2017. Bayes Forest: a data-intensive generator of morphological tree clones. *GigaScience*. Accepted.

Preston, R. 2006. Tall for its age. *The New Yorker*, P. 32–36.

Preuksakarn, C., Boudon, F., Ferraro, P., Durand, J.-B., Nikinmaa, E., and Godin, C. 2010. Reconstructing plant architecture from 3D laser scanner data. P. 12–17 in *6th International Workshop on Functional-Structural Plant Models*.

Puttonen, E., Suomalainen, J., Hakala, T., Räikkönen, E., Kaartinen, H., Kaasalainen, S., and Litkey, P. 2010. Tree species classification from fused active hyperspectral reflectance and LIDAR measurements. *Forest Ecology and Management*, 260(10):1843 – 1852.

Raumonen, P., Casella, E., Calders, K., Murphy, S., Åkerblom, M., and Kaasalainen, M. 2015. Massive-scale tree modelling from TLS data. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3/W4:189–196.

Raumonen, P., Ideguchi, Y., Uranishi, Y., Åkerblom, M., Kaasalainen, M., Yoshimoto, S., Kuroda, Y., and Oshiro, O. 2016. Virtual reality forest: Realistic trees based on laser scans. P. 5 pages in *Proceedings of EuroVR 2016*.

Raumonen, P., Kaasalainen, S., Kaasalainen, M., and Kaartinen, H. 2011. Approximation of volume and branch size distribution of trees from laser scanner data. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(5/W12).

Sarmiento, A. L., Bartholomeus, H., Herold, M., Martius, C., Malhi, Y., Bentley, L. P., Shenkin, A., and Raumonen, P. 2015. Application of terrestrial LiDAR and modelling of tree branching structure for plant-scaling models in tropical forest trees. P. 3–3 in *Proceedings of the SilviLaser 2015 conference*.

Shinozaki, K., Yoda, K., Hozumi, K., and Kira, T. 1964a. A Quantitative Analysis of Plant Form–The Pipe Model Theory I. Basic Analyses. *Japanese Journal of Ecology*, 14(3):97–105.

Shinozaki, K., Yoda, K., Hozumi, K., and Kira, T. 1964b. A Quantitative Analysis of Plant Form–The Pipe Model Theory II. Further Evidence of the Theory and its Application in Forest Ecology. *Japanese Journal of Ecology*, 14(4):133–139.

Shlyakhter, I., Rozenoer, M., Dorsey, J., and Teller, S. 2001. Reconstructing 3D tree models from instrumented photographs. *Computer Graphics and Applications, IEEE*, 21(3):53–61.

Smith, A. L., Astrup, R., Raumonen, P., Liski, J., Krooks, A., Kaasalainen, S., Åkerblom, M., and Kaasalainen, M. 2014. Tree root system characterization and volume estimation by terrestrial laser scanning and quantitative structure modeling. *Forests*, 5(12):3274–3294.

Stysley, P. R., Coyle, D. B., Kay, R. B., Frederickson, R., Poulios, D., Cory, K., and Clarke, G. 2015. Long term performance of the high output maximum efficiency resonator (HOMER) laser for NASA's global ecosystem dynamics investigation (GEDI) lidar. *Optics & Laser Technology*, 68:67–72.

Teng, C.-H., Chen, Y.-S., and Hsu, W.-H. 2007. Constructing a 3D trunk model from two images. *Graphical models*, 69(1):33–56.

Thies, M. and Spiecker, H. 2004. Evaluation and future prospects of terrestrial laser scanning for standardized forest inventories. *Forest*, 2(2.2):1.

Vonderach, C., Voegtle, T., and Adler, P. 2012. Voxel-base approach for estimating urban tree volume from terrestrial laser scanning data. *Int. Arch. Photogr. Remote Sens. Spat. Inf. Sci*, P. 39–B8.

West, G. B., Brown, J. H., and Enquist, B. J. 1999. A general model for the structure and allometry of plant vascular systems. *Nature*, 400(6745):664–667.

Woodgate, W., Armston, J. D., Disney, M., Suarez, L., Jones, S. D., Hill, M. J., Wilkes, P., and Soto-Berelov, M. 2017. Validating canopy clumping retrieval methods using hemispherical photography in a simulated Eucalypt forest. *Agricultural and Forest Meteorology*, 247(Supplement C):181 – 193.

Xu, H., Gossett, N., and Chen, B. 2007. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Transactions on Graphics (TOG)*, 26(4):19.

Yan, D.-M., Wintz, J., Mourrain, B., Wang, W., Boudon, F., and Godin, C. 2009. Efficient and robust tree model reconstruction from laser scanned data points. P. 572–576 in *Proceedings of the 11th IEEE International conference on Computer-Aided Design and Computer Graphics*.

Zheng, G. and Moskal, L. M. 2012. Leaf Orientation Retrieval From Terrestrial Laser Scanning (TLS) Data. *IEEE Transactions on Geoscience and Remote Sensing*, 50(10):3970–3979.

Publications

This chapter contains the complete included publications. Each individual publication is preceded by a page containing the publication details. The QR code on these pages contain the respective DOI address for convenience.

Article I

Comprehensive quantitative tree models from TLS data

Åkerblom, M., Raumonen, P., Kaasalainen, M.,
Kaasalainen, S., and Kaartinen, H.

IEEE International Geoscience and Remote Sensing Symposium Proceedings

DOI: https://doi.org/10.1109/IGARSS.2012.6352751

2012

COMPREHENSIVE QUANTITATIVE TREE MODELS FROM TLS DATA

Markku Åkerblom, Pasi Raumonen
and Mikko Kaasalainen

Tampere University of Technology
Department of Mathematics
P.O. Box 553, FI-33101 Tampere, Finland

Sanna Kaasalainen and Harri Kaartinen

Finnish Geodetic Institute
Remote Sensing and Photogrammetry
P.O.Box 15, FI-02431 Masala, Finland

ABSTRACT

We present comprehensive and quantitative tree models reconstructed from terrestrial laser scanning data. The tree models consist of large number of cylinders whose location, size, and orientation locally approximate the geometry of the tree. The parent-child relations of the cylinders also record the topological branching structure of the tree. The modeling process is automatic and scale-independent. When the tree model is computed once, it can be used to compute tree attributes, such as branch size distributions and taper functions, without the need to revisit the original dataset. The model is also compact, achieving a hundred- to thousandfold data size compression compared to the original dataset. We present also a validation of the model using generated tree models and examples of models from measurements of actual trees.

Index Terms— Tree model, terrestrial laser scanning, surface reconstruction, cylinder model, tree attributes

1. INTRODUCTION

Comprehensive and quantitative data-fitted tree models can be valuable tools for many forest-related fields, such as forest and environmental research, lumber industry, and overall forest measuring. These fields are interested in various geometric and topological tree properties, and have developed multiple ways to analyse these from terrestrial laser scanner (TLS) measurements [1, 2]. Many attribute-specific models exist in which the starting point is some or few geometric or topological tree attributes which are estimated [5, 6, 7]. The use of multiple models to acquire different attributes requires repeated analysis of the original TLS dataset which takes computational time and memory.

We propose an approach where we first reconstruct a comprehensive tree model (see Fig. 1) from which any of these attributes can be computed at will. Furthermore, after the reconstruction of the model, there is no need to revisit the TLS dataset as all the information is stored in the model as a compact presentation. For example the following attributes are easily accessible from the model: total and partial volumes, branch size distribution, bifurcation frequency and angles, trunk and branch profiles, etc.

Our modeling scheme [3, 4] is comprehensive as it resolves all the visible parts of the tree and even interpolates accurate estimates for some parts not present in the dataset.

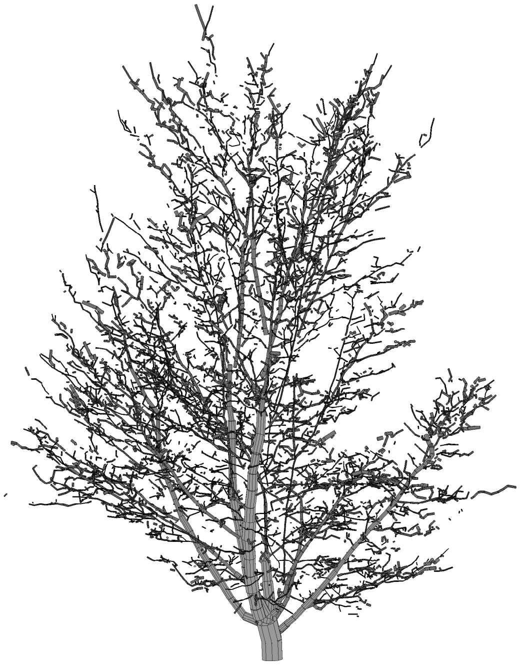


Fig. 1: A cylinder model of a broadleaf tree. The model was reconstructed from a point cloud with 1.8 million points, and it consists of 9200 cylinders.

This study was financially supported by the Academy of Finland projects: *Modelling and applications of stochastic and regular surfaces in inverse problems* and *New techniques in active remote sensing: hyperspectral laser in environmental change detection*.

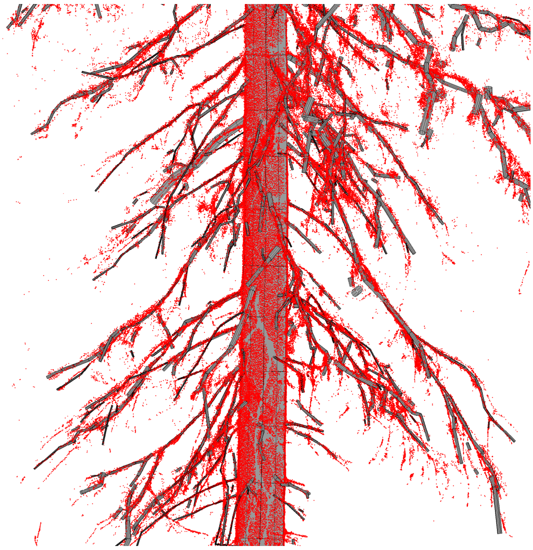


Fig. 2: Close-up of the cylinder model and point cloud of a Norway spruce. The reconstructed cylinder model contains some 6500 cylinders. The measurements have been thinned out for visualization.

The trunk and branches are presented quantitatively as a set of cylinders whose location, orientation, and size accurately approximate the tree locally. In addition, the branching structure is stored in the model, giving the topology of the tree. The model can be reconstructed from a dataset automatically without user interaction. This allows the modeling scheme to be used, e.g., in a batch process with a large number of trees. Because the all the essential information sought from the measurements can be calculated from the model, which is hundred- to thousandfold more compact in size, the model offers a save in memory requirements.

The tree model reconstruction requires only a few assumptions to be made about the TLS dataset and the tree. First, the TLS measurements should be from multiple positions (3 or more) around the tree, in order to get a comprehensive cover, and combined into a single point cloud. Second, we assume that the point cloud is an extensive enough and locally uniform sample of the tree surface embedded in the 3D Euclidean space. Third, the dataset must be coverable with large enough neighbourhoods that conform to the local details of the surface. Fourth, the order of magnitude of the branch and trunk size to set the reference scale and the approximate trunk direction are assumed to be known. Finally, the tree must be locally approximately cylindrical for the cylinder model to be useful.

Next the outline of the modeling scheme is described in more detail. Later the process is validated using generated tree models, and examples of tree models reconstructed from real measurements are shown. Example tree attributes are also computed. For more comprehensive details and validation, see [8].

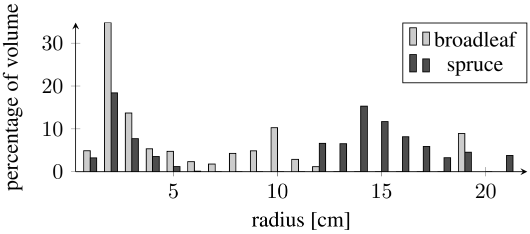


Fig. 3: Branch size distributions computed from the cylinder models shown in Figs. 1 and 2.

2. TREE MODELING PROCESS OVERVIEW

The basic idea of the model reconstruction process is to build the global tree model, which is a priori unknown, from the local details of the dataset, which are more easily determined. At the heart of this process are covers which consist of small sets along the tree surface and approximate its local details. The neighbour relations of these cover sets enable us to extend given sets along the surface and thus change in scale from local details to more global ones. The geometric properties of the cover sets approximate the local properties, such as the size, shape, and orientation, of the surface. Most of the process takes place in the topological level of the cover sets, the number of which is considerably smaller than that of points in the dataset. This makes the process faster in computational time and lowers the memory requirements.

The first step of the modeling process is to generate a cover of the point cloud with small neighbourhoods conforming to the local details of the surface. The cover sets are balls whose radius is about the same as the radius of the smallest branches. Each point of the dataset belongs in at least one of the cover sets and since the sets are intersecting, their neighbour-relation is easily determined. The neighbour-relation allows the determination of the connected components of the datasets.

The cover sets are classified using their geometric properties, such as their dimensionality and direction, to find the trunk of the tree and its base. After that the measurements from the surrounding ground, the undergrowth and possible other trees are filtered out using the neighbour relation of the cover sets. The remaining cover sets are divided into *components* based on their connectedness and possible components not part of the tree are removed using heuristics. Ideally, each dataset contains only one component, but imperfections in the data often lead to a large number of components.

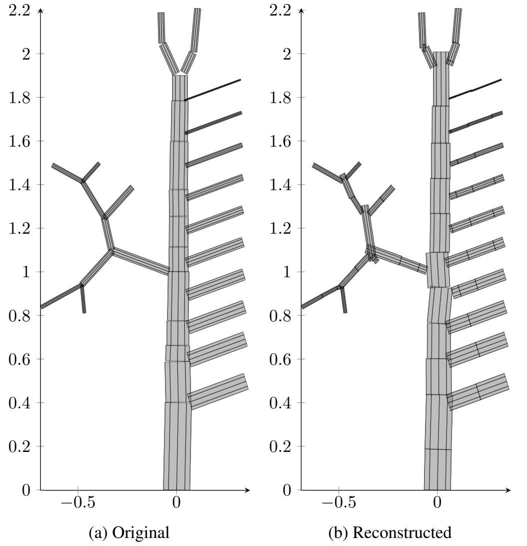


Fig. 4: Comparison of the original (left) and the reconstructed (right) cylinder models.

Each component is segmented into connected parts with no bifurcations such that each segment corresponds to a branch or part of a branch of the tree. The segmentation process is based on local recognition of bifurcations, which in turn is based on the connectivity. The neighbour-relation is used for set extension which is the main tool. The segmentation process starts from the base of a component and during the process a small region, consisting of cover sets, is moved along the component step by step using the neighbour information. Then at each step the connectivity of this region is checked, and in the cases of multiple connected components, possible bifurcations are detected. The proposed bifurcation part of the current segment is analysed further to know, whether the part will end or continues to expand, and if it is a new branch or continuation of the current segment. The bases for new child segments are recorded, and segments are created one-by-one until all the components have been processed. Branching structure information is recorded during the segment forming process, and transferred later to the cylinder model.

Since the branches can be curved and have a non-constant radius, each segment is further divided into even smaller parts, called *regions*. Each region should be nearly straight and have nearly constant radius, so that it can be accurately approximated by a cylinder.

A cylinder is fitted to the points of a region as a least-squares minimization problem. Geometric properties of the region, approximated from its points, are used to find initial values for the fitting process. A priori information, e.g., branch radius decreases when moving outwards away from the trunk, is also used to find the best local reconstructions.

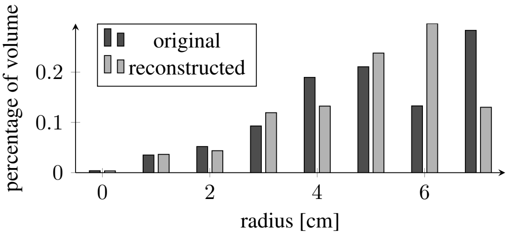


Fig. 5: Branch size distribution of the original and the reconstructed tree models.

When cylinders have been fitted to all the segments, the cylinder model can be further refined by locating and filling gaps between cylinders. The relational information of the cylinders is used to identify the possible gaps in the cylinder model. With the added cylinders the tree model is complete.

Figs. 1 and 2 show examples of completed cylinder model reconstructions for a broadleaf tree and a Norway spruce, respectively. Fig. 3 shows the computed branch volume distributions for the trees.

3. VALIDATION WITH GENERATED TREE MODELS

It is hard to validate the modeling scheme by using real trees as their branching structure and branch sizes are practically impossible to measure for whole trees. Therefore, we use a simple generated cylindrical tree model to get some quantitative validation results. This way the detailed structure of the sampled original tree is known, and it can be compared to the reconstructed one. Error estimation for the reconstruction is simple and accurate.

The sampled original tree is defined as a set of cylinders, and then, to simulate laser scanning, random samples are taken from the surface of each cylinder. Here we do not try to simulate real laser scanning as closely as possible, but simply generate a random sample points with uniform distribution from the surfaces of the cylinders; i.e., the sample density is kept constant all over the tree. Then we add Gaussian measurement noise to each sample point in the direction normal to the cylinders so that the points generally do not lie on the cylinders. An example of a generated tree model is shown in Fig. 4a.

Reconstruction level for the given generated model has been studied in detail in [8], by changing the level of the measurement noise, and the sample density. The study is done without units, but if the units are considered as meters (branch radius from 3mm to 7cm), the following values can be found. The results show that close to complete reconstruction is possible even when the measurement noise is in the interval [-9mm, 9mm]. The measurement noise of the used laser scanner is close to the diameter of the beam, which is 2–3mm. There are, of course, other more unpredictable error sources, such as wind conditions which may increase the overall noise level. With sample densities above about 0.3 points/cm^2 the reconstruction is possible. Since the point density of the measurements for the tree in Fig. 1, approximated using the reconstructed cylinder model, is about 2.3 points/cm^2, the theoretical lower limit can be easily met with TLS.

Fig. 4b shows the reconstructed tree model when measurement noise level is 3mm and sample density 0.852 points/cm^2. These are estimated values for real laser scanning measurements. The cover set radius is 2.16cm. The difference of the total volumes of the original and the reconstructed models is 7.4%. The difference in the total length of the cylinders is 1.0%. The branch volume distributions for both models are shown in Fig. 5.

The visualization, error percentages and the branch volume distributions show that the reconstruction is very good. Even the smallest branches are successfully segmented and reconstructed. The branch volume distribution shows that parts of the trunk are reconstructed smaller than they should (differences in the 6cm and 7cm bars). This is due to the subtle overlapping of the trunk and the large number of bifurcating branches.

4. RESULTS AND DISCUSSION

The presented tree modeling scheme produces accurate 3D cylinder models from TLS data. The approach is both quantitative and comprehensive as generated and real measurement examples have shown. The produced cylinder model contains the properties of the cylinders approximating the tree locally. In addition the relations between the cylinders are known as is the information which cylinders form a single branch. The models are easy and fast to visualize, and information, such as bifurcation angles and frequencies, and size distributions, is easily extracted.

The reconstruction is automatic; a few input parameters for cover specification and possible filtering are required, but that is the extent of user interaction during the model production. Since the method is based on topological concepts, such as connectivity, it is scale-independent. Thus, the same realization of the method can be used to analyse tree-like point clouds of various sizes, and the accuracy of the method is mainly restricted by the measurements.

Because only the cover sets and their center points and neighbour-relation are needed for most of the steps, only tens or hundreds of thousands of "points" are often used instead of all the millions of measured points. Thus, the memory requirements and the computational time are greatly reduced. We have implemented the method in MATLAB and the analysis of a single tree takes somewhere from minutes to ten minutes (2.8GHz Intel Core i7, 8GB RAM).

The development of the algorithm and its implementation continues actively. The goal is to make the modeling process completely automatic by choosing the cover set radius based on, e.g., the estimated sample density. Currently, the modeling scheme is being adapted for tree root modeling as well. We are also planning to use elliptic cylinders and cones instead of circular cylinders.

5. REFERENCES

[1] C. Schütt, T. Aschoff, D. Winterhalder, M. Thies, U. Kretschmer, and H. Spiecker, "Approaches for recognition of wood quality of standing trees based on terrestrial laserscanner data," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. Part 8, pp. W2, 2004.

[2] 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. Part 8, pp. W2, 2004.

[3] P. Raumonen, S. Kaasalainen, M. Kaasalainen, and H. Kaartinen, "Approximation of volume and branch size distribution of trees from laser scanner data.," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 38, no. 5/W12, 2011.

[4] P. Raumonen, M. Kaasalainen, M. Åkerblom, S. Kaasalainen, H. Kaartinen, M. Vastaranta, and M. Holopainen, "Comprehensive quantitative tree models from terrestrial laser scanner data," *Canadian Journal of Forest Research*, submitted.

[5] J.F. Côté, J.L. Widlowski, R.A. Fournier, and M.M. Verstraete, "The structural and radiative consistency of three-dimensional tree reconstructions from terrestrial lidar," *Remote Sensing of Environment*, vol. 113, no. 5, pp. 1067–1081, 2009.

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

[7] N. Pfeifer, B. Gorte, and D. Winterhalder, "Automatic reconstruction of single trees from terrestrial laser scanner data," *Information Sciences*, vol. 35, no. Part B, pp. 114–119, 2004.

[8] M. Åkerblom, "Quantitative tree modeling from laser scanning data," M.S. thesis, Tampere University of Technology, 2012.

Article II

Fast automatic precision tree models from terrestrial laser scanner data

Raumonen, P., Kaasalainen, M., Åkerblom, M., Kaasalainen, S., Kaartinen, H., Vastaranta, M., Holopainen, M., Disney, M., and Lewis, P

Remote Sensing, 5(2):491–520.

DOI: http://dx.doi.org/10.3390/rs5020491

2013

Remote Sens. **2013**, *5*, 491-520; doi:10.3390/rs5020491

OPEN ACCESS

Remote Sensing

ISSN 2072-4292

www.mdpi.com/journal/remotesensing

Article

Fast Automatic Precision Tree Models from Terrestrial Laser Scanner Data

Pasi Raumonen [1,*], Mikko Kaasalainen [1], Markku Åkerblom [1], Sanna Kaasalainen [2], Harri Kaartinen [2], Mikko Vastaranta [3], Markus Holopainen [3], Mathias Disney [4] and Philip Lewis [4]

[1] Department of Mathematics, Tampere University of Technology, P.O. Box 553, FI-33101 Tampere, Finland; E-Mails: mikko.kaasalainen@tut.fi (M.K.); markku.akerblom@tut.fi (M.Å.)

[2] Remote Sensing and Photogrammetry, Finnish Geodetic Institute, P.O. Box 15, FI-02431 Masala, Finland; E-Mails: sanna.kaasalainen@fgi.fi (S.K.); harri.kaartinen@fgi.fi (H.K.)

[3] Department of Forest Sciences, University of Helsinki, P.O. Box 27, FI-00014 Helsinki, Finland; E-Mails: mikko.vastaranta@helsinki.fi (M.V.); markus.holopainen@helsinki.fi (M.H.)

[4] Department of Geography, University College London, Pearson Building, Gower Street, London WC1E 6BT, UK; E-Mails: mathias.disney@ucl.ac.uk (M.D.); ucfalew@ucl.ac.uk (P.L.)

* Author to whom correspondence should be addressed; E-Mail: pasi.raumonen@tut.fi.

Received: 3 November 2012; in revised form: 24 December 2012 / Accepted: 15 January 2013 / Published: 25 January 2013

Abstract: This paper presents a new method for constructing quickly and automatically precision tree models from point clouds of the trunk and branches obtained by terrestrial laser scanning. The input of the method is a point cloud of a single tree scanned from multiple positions. The surface of the visible parts of the tree is robustly reconstructed by making a flexible cylinder model of the tree. The thorough quantitative model records also the topological branching structure. In this paper, every major step of the whole model reconstruction process, from the input to the finished model, is presented in detail. The model is constructed by a local approach in which the point cloud is covered with small sets corresponding to connected surface patches in the tree surface. The neighbor-relations and geometrical properties of these cover sets are used to reconstruct the details of the tree and, step by step, the whole tree. The point cloud and the sets are segmented into branches, after which the branches are modeled as collections of cylinders. From the model, the branching structure and size properties, such as volume and branch size distributions, for the whole tree or some of its parts, can be approximated. The approach is validated using both measured and modeled terrestrial laser scanner data from real trees and detailed 3D models. The results show that the method allows an easy extraction of various tree attributes from terrestrial or mobile laser scanning point clouds.

Keywords: terrestrial laser scanning; automatic tree modeling; precision tree models; segmentation; forest inventory; branch size distribution; carbon cycle estimation

1. Introduction

The determination and prediction of tree characteristics and quality attributes is important in forest management, especially in pre-harvest measurements [1]. These attributes are geometric and statistical characteristics of trees such as the crown-base height, total above-ground volume, the branch size distribution, and the branching structure. In particular, timber assortments, tree quality, branch decay times and carbon cycle estimations, *etc*. require accurate estimates on branch sizes and other tree attributes. However, many of these characteristics have been difficult or even impossible to measure operationally, often requiring cutting and laborious manual measurements.

One way to estimate parameters that are hard to measure, particularly for a group of similar trees, is to use other simpler measures together with statistical models developed from detailed manual measurements. For example, the biomass of a tree can be estimated quickly and quite accurately from stem diameter at breast height and height of the tree [2]. Traditionally, the crown–base height is measured and used in estimation of timber quality. However, there are obvious limitations on the accuracy and applicability of such simplified statistical models.

With accurate and fast-to-use laser scanners, tree parameters can be determined with fewer practical difficulties [3]. 3D mapping of smaller areas with high detail is possible with terrestrial laser scanning (TLS) which can produce dense 3D point clouds of the tree surfaces [4]. However, often the laser scanner data are only used together with statistical models to give statistical estimates of tree properties. For example, standing tree biomass and its changes can be measured with TLS because they are highly correlated with the number of hits in the TLS point cloud [5,6].

TLS can produce 3D point clouds that allow the quantitative analysis of trees and the reconstruction of quantitative 3D tree models. These are required in forest research in general, and they can be also used to develop better statistical models for tree attributes. ¿From a TLS point cloud, one can measure tree and stand characteristics such as the location, height, crown coverage, species and stem curve [6–12]. In forest research, TLS has been used for the detailed modeling of individual trees and canopies [7,8,11,13,14]. However, a typical property of the modeling procedures has been the estimation of only a few attributes from TLS data, and only for a limited part of the tree, e.g., stem length/volume.

Many problems in forestry, biomass estimation, forest research, and forest remote sensing could be more readily tackled if it were routinely possible to fit tree models to TLS measurements, such that a model is:

1. *Comprehensive*: it (i) covers all parts of the tree that are resolved in the data; and; (ii) interpolates to obtain credible reconstructions of the unseen parts that are not represented in the dataset but are located between some seen parts.
2. *Precise*: its parts are best-fit solutions describing the corresponding parts of the tree accurately and in detail, giving their location, size, orientation, relation to other parts of tree, or many other desired topological and metric attributes.
3. *Compact*: it is easily stored and manageable, and any attributes can be readily extracted from it anytime after its construction without having to use the actual data. Most of the relevant information of the original data is retained for future use in a compact form, even if one does not yet know what that information is.
4. *Automatic*: it is constructed without manual operation such that the pipeline processing of a large number of measured trees is possible.
5. *Fast*: a single tree can be modeled virtually immediately (within minutes).

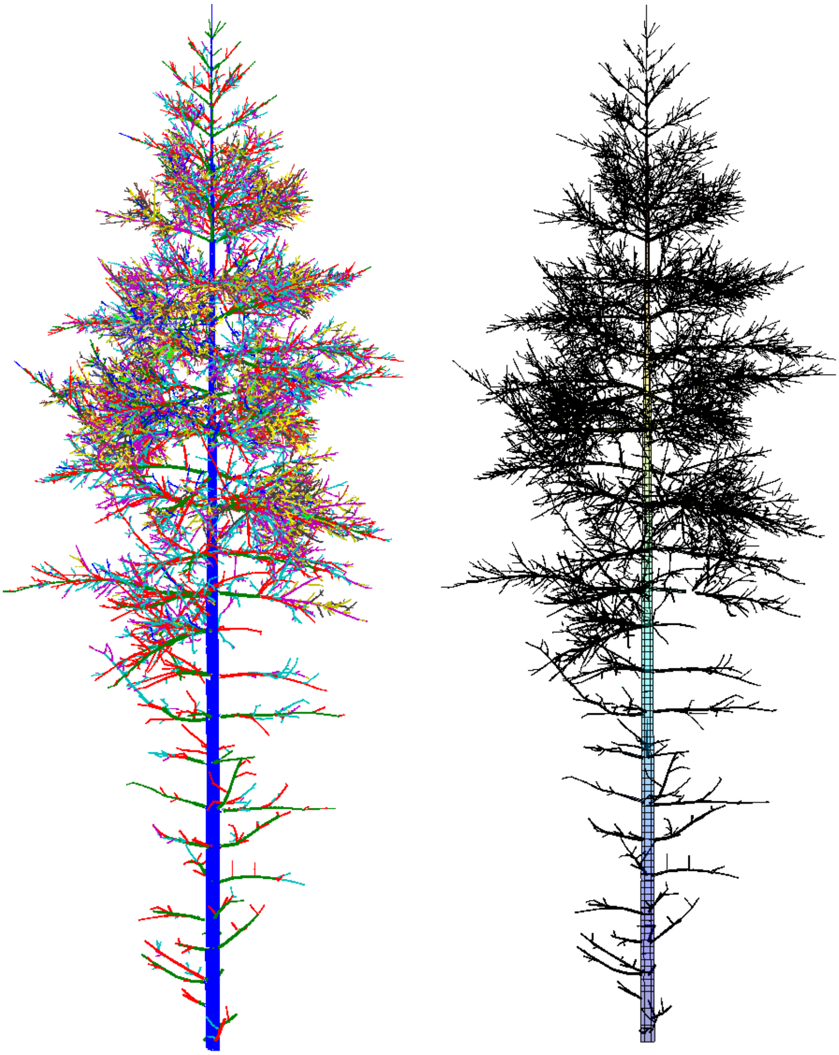
Modeling procedures fulfilling all the conditions above have hitherto not been available, although various methods have been developed to allow automated tree reconstruction [10,15–20].

In this paper, we present a new method that produces 3D tree models from TLS data fulfilling all the conditions above (see the examples of reconstructed models in Figures 1 and 2). In particular, we present all the modeling steps from the point cloud data to a complete tree model. One of the core ideas behind the approach introduced here is that practically any external attributes of a tree can be approximated accurately at will from a compact model of the type above. The attributes can be, e.g., the volume and its distribution along different parts of the tree; the lengths and taper functions for the trunk and branches; the bifurcation frequency, topology and branching angles, a 3D "branch map", *etc*. In this paper, we restrict the models to cover the woody parts of the tree (no foliage or needles), and only consider scans from a single tree.

Our scheme is based on the principle of building the global model step by step by an advancing collection of small connected surface patches. These patches are small local subsets of the point cloud and their geometric properties and neighbors are easily defined. This building-brick approach makes the method robust as the procedure does not need to know what a tree is supposed to look like and the point density can be quite varying. An ordered collection of local connected surface patches automatically yields the global structure both qualitatively and quantitatively.

A key part of any modeling method is the segmentation of the point cloud into branches. The segmentation gives the topological tree structure and the resulting segments (branches) can be then geometrically reconstructed. Our segmentation procedure uses the surface patches and recognizes bifurcations along the tree surface by checking local connectivity of a moving surface region. Other methods for segmenting have been presented: one way is to use voxels and mathematical morphology [21]. Another is to use octree based skeletonization approach [22,23]. Skeletons can be defined also using a neighborhood graph and checking the connected components of the level sets of the graph [17,24].

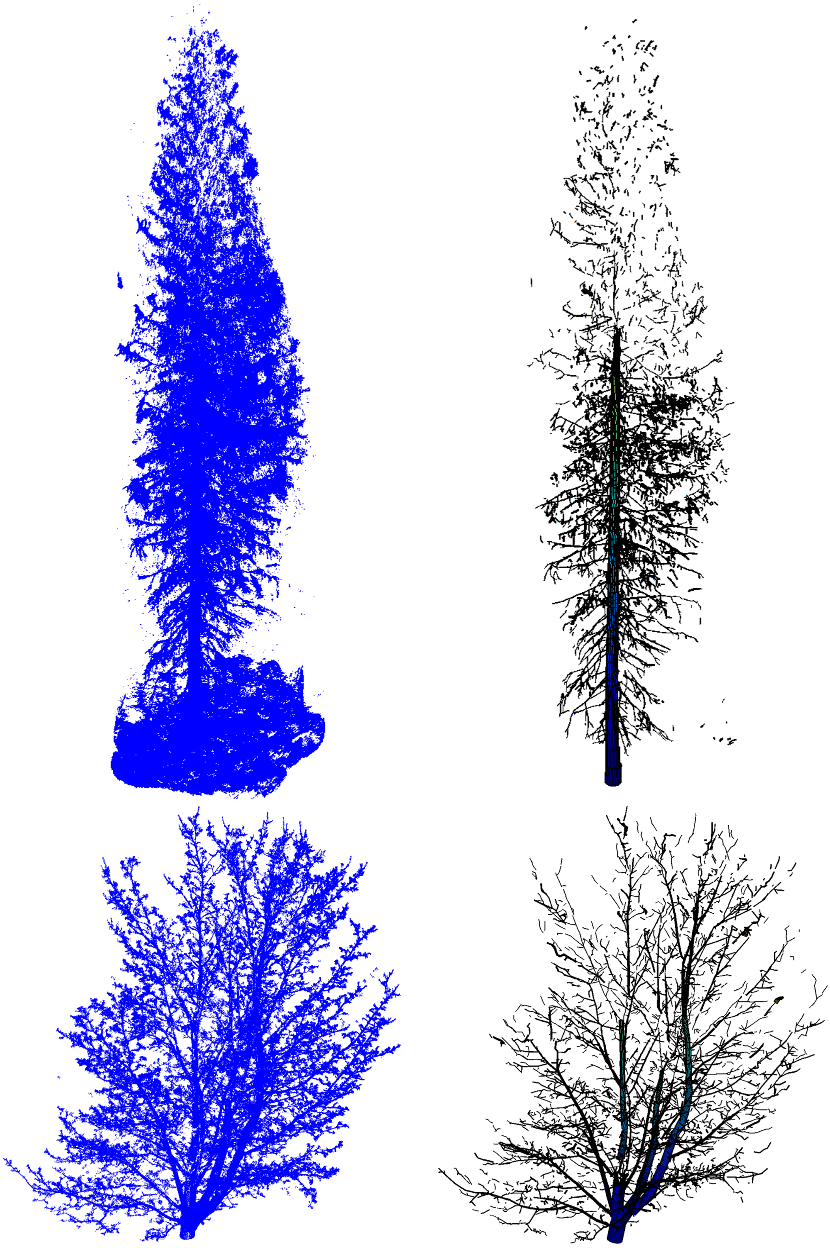
Figure 1. Segmented point cloud (**left**) and the final cylinder model (**right**) of the artificial Scots pine.



The paper is organized as follows. The method is presented in Section 2; in Section 2.1 we describe our data, and the outline of the method is given in Section 2.2. In Sections 2.3–2.10 we describe the method in detail, with some algorithmic points covered in the Appendix. In Section 3 we test the method using real and artificial TLS point clouds and show some results. Finally, Sections 4 and 5 contain discussion and conclusions.

We have published two short conference papers [25,26] describing some of the ideas of our method. In this paper, however, we develop the method further and present many more details, tests, and validations. The primary novel features of our method are: (i) the partition of the point cloud into patchlike sets that allow a fast automatic procedure; (ii) efficient segmentation rules that retain the topological and hierarchical information; and (iii) a tool-like interface with direct handles for any geometric and topological attributes of the tree. We render the processed point cloud of a tree in a readily accessible geometric mode that can be utilized by a wide variety of applications and end-users.

Figure 2. Point clouds and their final cylinder models. The point cloud (**top left**) and the cylinder model with some 7,070 cylinders (**top right**) of the spruce. The point cloud (**bottom left**) and the cylinder model with about 6,820 cylinders (**bottom right**) of the maple.



2. Method

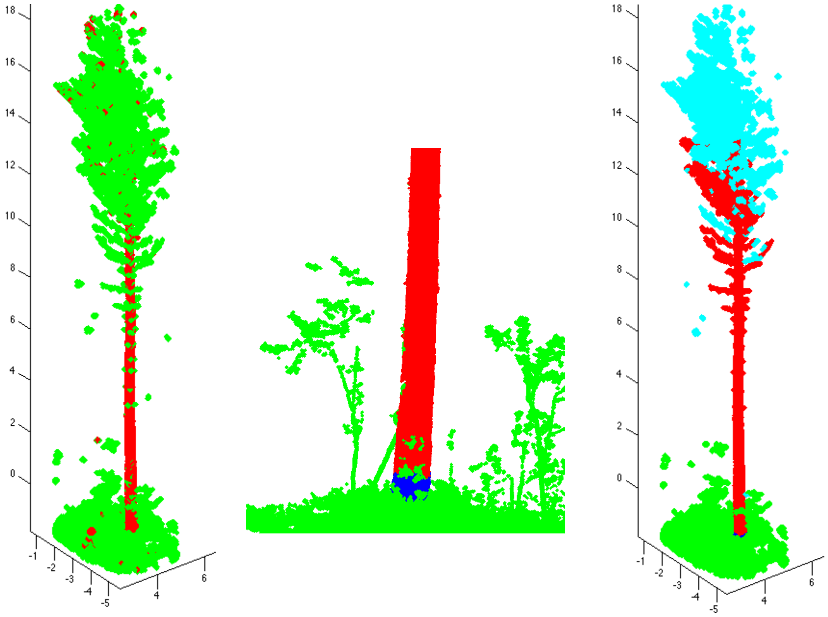
2.1. Data: Point Clouds

A (laser scanner produced) *point cloud* P_M is a finite subset of the 3-dimensional real coordinate space $\mathbb{R}^3$. In this paper, each point $\mathbf{x} \in P_M$ gives the Cartesian x, y, z-coordinates of the corresponding scanned point. Furthermore, we assume that each point cloud P_M is a sample of a surface M embedded in $\mathbb{R}^3$. The surface M represents the surface of the scanned tree, and the point cloud PM is a finite but dense enough sample of M such that the point density varies little within distances of about the maximum trunk diameter. Notice also that the points in P_M are unorganized and can be saved in any order as rows into a three-column matrix.

In this paper, we have used scans from four trees: one Norway spruce [*Picea abies* (L.) H. Karst] (see Figure 2), two Scots pines [*Pinus sylvestris*] (see the other in Figure 3), and one Norway maple [*Acer platanoides*] (see Figure 2). The trees were scanned from three different directions to have a comprehensive cover of the branching structure. The scans were registered to a common coordinate system via spherical reference targets placed in the measured area. Our method handles the TLS data only as point clouds and does not take into account the particular features of the equipment used. For the datasets used in this paper, our system consisted of a phase-based terrestrial laser scanner (Leica HDS6100 with a 650–690 nm wavelength). The distance measurement accuracy of the scanner is 2–3 mm, and the field-of-view is $360° \times 310°$, the circular beam diameter at the exit is 3 mm and the beam divergence is 0.22 mrad (see [9,27] for more details). The measured point clouds for the trees we have studied in this paper contain roughly one to five million points each. The horizontal distance of the scanner to the trunks and the point separation angle were about 7–12 m and 0.036 degrees, respectively. Thus the average point density on the surface of the trunk (at the level of the scanner) for a single scan is about 2–5 points per square centimeter.

We have also used a complex artificial tree model and simulated laser scanning to produce point clouds. This was done for controlled testing and validation of the method. The 3D structural tree model used here represents a 30-year-old Scots pine tree (see Figure 1 in this paper and Figure 4 in [28]). The model is generated using an empirical growth model parametrized by species-dependent branching statistics in conjunction with specified external environmental conditions [29]. TLS point clouds were simulated using the Monte Carlo ray-tracing code which has been used for a wide range of applications [28,30–32]. Simulation parameters were the same as in the scanner used in the measurements (see above). The simulated lidar was situated at 1.5 m height, at a radial distance of 20 m from the tree (at [0, 0, 0]), at 70°, 0° and –70° in the xy plane, *i.e.*, xyz locations [0, 20, 1.5], [18.7939, 6.8404, 1.5] and [–18.7939, 6.8404, 1.5]. The simulated point cloud has about 4.5 million points.

Figure 3. Determination of tree components and their bases. Left: The initial classification of trunk points (red). Middle: Part of the trunk point set (red) and its base (blue). The green denotes sets not part of the tree. Right: Final classification of components. The component starting from the base (red), the other tree components (cyan), and the points not part of the tree (green). The units of the axes in the left and right figure are meters.



2.2. Outline of the Method

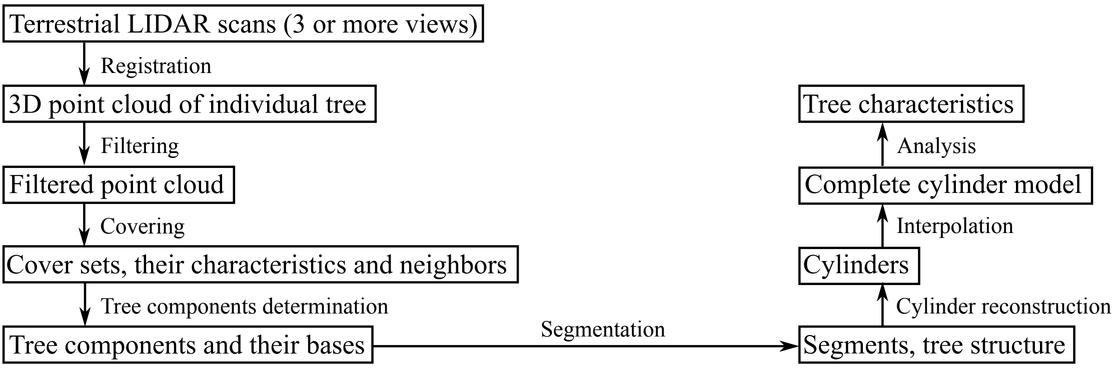
Our method is based on covering the point cloud data with small sets corresponding connected patches on the tree surface. Then using a building-brick approach, the unknown global tree model is built by "growing" the tree surface step-by-step using the surface patches. After the patchwork over the tree has been obtained and the tree has been segmented into its branches, the remaining question is how to render it in a form suitable for fast computations and storage. The choice adopted here is the simplest one: the tree surface is reconstructed as an aggregate of small cylinders of varying size approximating parts of the trunk and branches. Cylinders (or their generalizations) are essentially a robust regularization choice that is sufficient for providing the essential attributes of the tree such as stem and branch diameters.

All the desired external characteristics of the tree can be readily approximated from the cylinder data. Moreover, the method is scale-independent because it uses only topological properties and relative sizes. Absolute size restrictions are related to cover set size (building brick size) and the accuracy and density of measurements. Thus, if the cover set size is small enough, the method can be used to reconstruct the model accurately down to the measurement accuracy of the laser scanning. From the point of view of information compression, the cylinder model retains most of the information of the original point cloud in a format that is hundreds to thousand times more compressed in size.

Constructing the tree model requires some assumptions and *a priori* knowledge about the data and trees. First, we assume that the point cloud is a locally uniform and extensive enough sample of the surface in a 3D Euclidean space modeling of the real tree. Second, it must be possible to cover the point cloud sample with small sets that correspond to connected patches along the surface. Third, some other features of the tree, such as the order of magnitude of the branch and trunk size and the approximate trunk direction, are assumed to be known. Finally, we assume the tree to be locally approximately cylindrical. In the future versions of the algorithm, we will include the option of deformable cylindrical surfaces whenever this approximation is significantly violated.

The main steps of the method are the following (see Figure 4). At first, the point cloud is filtered to remove noise or isolated points (see Section 2.3). Then the filtered point cloud is covered with small sets conforming to the surface of the tree (Section 2.4). Next the neighbor-relation of the cover sets is defined (Section 2.5) and the sets are geometrically characterized (Section 2.6). The neighbor-relation determines the connectivity properties, and the geometric characterizations are used, e.g., for the classification of trunk points. Next, the sets that are not part of the tree, such as the ground sets, are removed, and the tree components and their bases are defined (Section 2.7). Here, and throughout the paper, by a *tree component* we mean an essentially separate part or cluster of the point cloud that can be, e.g., a single branch, a collection of branches, or even the whole tree. Following this, the tree components are segmented (Section 2.8). Throughout the paper, by *segment* we mean a connected non-bifurcated part of the tree, such as a branch or part of a branch or the trunk. The segmentation also gives the ordered information of the tree structure. In the segmentation process, we use surface growing, and bifurcations are recognized by checking local connectivity. The next step is to approximate each segment as a sequence of cylinders of possibly varying radius, length, and orientation (Section 2.9). To complete the cylinder model of the whole tree, gaps between cylinders are sought and filled with additional cylinders (Section 2.10). Finally, statistical and other characteristics of the tree can be computed from the completed cylinder model (Section 3.1).

Figure 4. The main steps of the method.


Terrestrial LIDAR scans (3 or more views)
Registration
3D point cloud of individual tree
Filtering
Filtered point cloud
Covering
Cover sets, their characteristics and neighbors
Tree components determination
Tree components and their bases
Segmentation
Segments, tree structure
Cylinder reconstruction
Cylinders
Interpolation
Complete cylinder model
Analysis
Tree characteristics

2.3. Filtering Noise from the Sample

The point cloud sample may contain outliers and noisy points caused by various reasons such as interference effects. Such points are not regarded as samples of the actual surface we want to reconstruct, and the first step in our method is to filter some noise from the point cloud. Our approach to filtering is based on the covers of the point cloud that are defined in the next section.

To remove separate single points or few-point clusters, we cover the point cloud with small balls and reject points that are only included in balls that contain fewer than some small number of points. A more comprehensive version of this filtering scheme is to define a small ball for each point and then reject the points whose balls contain too few points. The size of the balls and the number of points depend on the density of the points in the data. We have used balls whose radius is 1.5 cm and removed all the balls with less than three points.

Another, and possibly additional, filtering scheme to remove small separate parts of the point cloud uses covers of larger balls to determine the connected components of the point cloud. Then the points in components with too few cover sets are removed. We have used balls whose radius is 3 cm and removed all the components with less than tree balls.

The values of the filtering parameters and their effects on resulting tree models depend on the noise level and its distribution in the data, the geometry of the tree, *etc.*, so there is no simple selection rule for the parameter values. However, we have found that, in practice, experience guides the selection of the values quite robustly. The effects are mainly local and usually not very sensitive to the parameter values. Furthermore, the same parameter values work well for similar trees and scanner parameters. For future versions of the algorithm, we will study automated machine learning of the parameter values and their adaptive adjustment.

2.4. Cover Sets

The embedding space $\mathbb{R}^3$ is endowed with the usual Euclidean distance dis and thus Euclidean topology. The restriction dis_P of the distance dis to P_M gives the distances between the points of P_M, and thus also a metric topology for P_M. Notice that dis_P locally approximates the distance function of the surface M and it can thus be used to approximate the surface M from the sample P_M. A spherical neighborhood of radius r at $\mathbf{x}$, or simply *r-ball at* $\mathbf{x}$, is the subset $B(\mathbf{x}, r)$ of P_M consisting of all the points $\mathbf{y}$ that satisfy $dis_P(\mathbf{x}, \mathbf{y}) < r$. A *cover* $C = \{B_i\}$ of P_M is a collection of subsets $B_i \subset P_M$ such that each point of P_M belongs in at least one of the subsets B_i. A cover is a *partition* if each point belongs in exactly one of the subsets.

We aim to reconstruct the surface M from the sample P_M, but the "global shape" and structures of M are impossible to determine directly from the whole sample. However, locally the surface and its structures can be approximated well from the sample. Furthermore, the sample P_M is generally quite dense compared to the size of the local details of M. Therefore, a much smaller subsample of P_M retains all the necessary information to reconstruct a cylinder model of the tree surface M. To accomplish these aims, we cover the sample P_M with such small sets that they are expected to correspond to connected subsets of the surface (see Figure 5). Thus, the cover sets are small, connected surface patches and they define a new much smaller sample of M, which locally approximates the shape and topology of the surface. Because the cover sets are in a way the smallest unit sufficient to represent the sample P_M, the size of the sets defines the limit of the details of M that can be accurately approximated (see Figure 6).

Figure 5. A cover which is a partition. Different colors denote different cover sets.

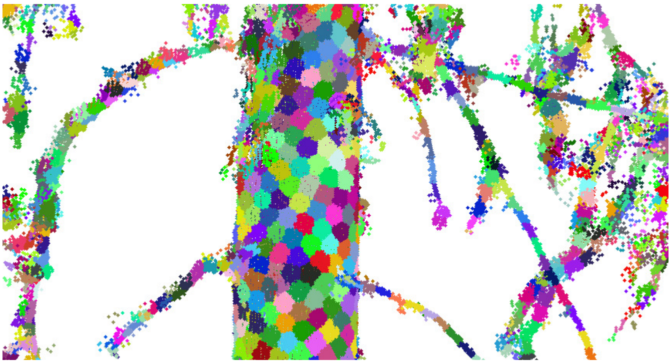
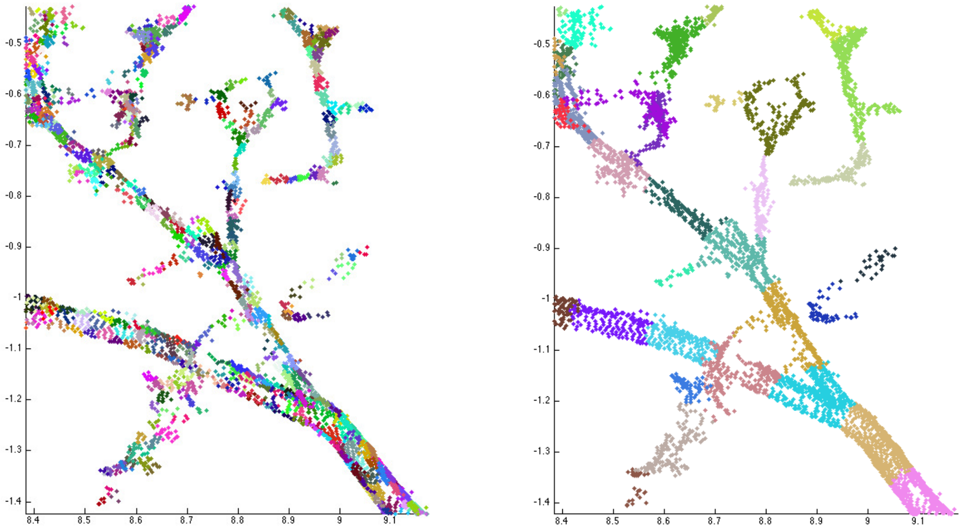


Figure 6. Comparison of the covers of a branch. The minimum diameters (d) of the cover sets are 2 cm (**left**) and 10 cm (**right**). The smaller cover sets can capture much more detail.



We generate two mutually related covers which are used for different purposes. First, we generate a cover $C_B = \{B_i\}$ of r-balls, which will then induce the other cover, a partition $C_P = \{b_i\}$. The cover C_B is used to approximate the structures of M and to define the neighbor-relation for the partition C_P, which is used to define the components and segments of the tree. The cover C_B of r-balls is random, but to distribute the balls evenly along the surface, there are two restrictions: (1) the minimum distance between the centers of two balls is d; and (2) the maximum distance from any point to the nearest center is also d. The parameter d is a little smaller than, or equal to, the radius r, and it controls the size of the cover sets in the partition C_P. The partition $C_P = \{b_i\}$ is induced by the r-balls: for each ball B_i there is a corresponding set b_i that consists of those points of B_i that are closer to the center of B_i than any other center. Thus the minimum diameter of the sets b_i, in the case they are not near, e.g., the tip of a branch, is d and the maximum possible diameter is $2d$.

The parameters d and r should be as small as possible so that the cover sets conform to the details of the surface. On the other hand, they should be large enough so that the sets can be reliably used to approximate different characteristics, such as surface normals, and to restrict the computational requirements. The parameters depend on the size of the smallest branches/details we are looking for, the point density, and the noise level. Usually, in the case of trees, d is about 1 to 3 cm. To generate the r-balls, we first partition P_M into cubes with side length r. Then each r-ball belongs to the r-cube containing the center and the 26 neighboring cubes.

2.5. Neighbor-Relation

The neighbor-relation for the partition C_P is a central tool defined by the r-balls: Let b_i and b_j be cover sets of the partition C_P, and B_i and B_j be the corresponding r-balls. Then b_i and b_j are *neighbors*, if either b_i and B_j or b_j and B_i have a common point. Thus, the smaller d is compared to r, the more neighbors there possibly are for each cover set in C_P. To guarantee that cover sets whose centers are up to $2d$ apart from each other are neighbors, as they should, r should be little larger than d. Notice also that the number of neighbors varies and thus a natural way to store the neighbor information is to use cell-arrays with variable cell size.

Let G be a cover set or a group of cover sets. With the neighbor-relations, G can be extended by adding its neighbors to it, and the resulting group of cover sets is denoted by $Ext(G)$. This can be repeated multiple times and we can, e.g., "grow" the tree surface from the given initial set and thus also change the scale from small to large.

One of the applications of the cover set extension is to determine connected components or connectedness of a given group of cover sets: two cover sets of C_P are in the same component if the cover sets can be expanded by the neighbor-relation to the same sets. Furthermore, the maximal disjoint sets generated by the cover set extension are the connected components of a given group of cover sets.

2.6. Characterization of Subsets

Next, we present a number of useful geometric characterizations of the cover sets and other subsets $B \subset P_M$ of the point cloud. Many of these use the eigenvectors and eigenvalues of the *structure tensor* (*i.e.*, scatter matrix or covariance matrix) $\mathbf{S}(B)$ of B [33]: Let $\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}$ be the unit eigenvectors of $\mathbf{S}(B)$ such that the corresponding eigenvalues satisfy $\lambda_1 \geq \lambda_2 \geq \lambda_3$.

If B is a r-ball, then $\{\mathbf{u}_1, \mathbf{u}_2\}$ span the tangent space and $\mathbf{u}_3$ approximates the *normal line* ($\mathbf{u}_3$ can point to either side of the surface M) [34]. The eigenvectors also give the *principal components* of the set B, *i.e.*, they describe the orthogonal components of the data exhibiting maximal variance [35]. The eigenvalues of $\mathbf{S}(B)$ can be used as indicators for the dimensionality of the set B [36], *i.e.*, if the set is elongated, planar or 3-dimensional.

We use the eigenvectors $\mathbf{S}(B)$ also to define the *branch direction line* $D(B)$ for each r-ball B as a unit vector that estimates the direction of the underlying branch or trunk at B. For clearly elongated sets we set $D(B)$ equal to $\mathbf{u}_1$. For other sets we use the normals of B and its neighboring sets to define

$D(B)$. Because branches are locally approximately cylindrical, the normals of the neighboring sets are all approximately orthogonal to the branch direction. Thus the vector that minimizes the sum of the squared dot products with the normals is a good approximation of the branch direction. Then $D(B)$ is the unit eigenvector corresponding to the smallest eigenvalue of the matrix $\mathbf{N}^T\mathbf{N}$, where $\mathbf{N}$ is the matrix whose rows are the unit normals of B and its neighbors.

For many trees, the trunk, at least near the base of the tree, is often almost straight and has generally a direction different from that of the most branches and the nearby ground. Thus, if the trunk direction can be estimated (globally) by a vector $\mathbf{T}$, then the angle between $\mathbf{T}$ and $D(B)$ estimates the *parallelism* of the underlying branch/trunk of a cover set B.

2.7. Tree Components and Their Bases

The next step, after the generation of a cover, the determination of its neighbor-relation, and the geometric characterization of its sets, is to extract the cover sets pertaining to the tree. In other words, the point cloud may contain ground and other points which are not part of the tree. These parts are removed and the base of the trunk is defined. Furthermore, because of possible gaps in the data, the point cloud may have multiple (often hundreds or thousands) connected components of the tree. The bases of these other tree components also need to be defined because later on, starting from the base each component is segmented into its branches. Although we next present an automatic way to separate the surroundings from the tree, we still assume that the tree is the only (big) tree in the point cloud. Hence the manual removal of adjacent trees from the point cloud may be needed. An automated way to handle multiple trees is a subject of future development.

To determine the tree components, the trunk is first defined approximately as a set **Trunk** of all those cover sets that are parallel to the approximated trunk direction and are two-dimensional (see also [9,25]). Next redefine the trunk by including its neighboring cover sets, *i.e.*, set $\mathbf{Trunk} = Ext(\mathbf{Trunk})$. On the left in Figure 3, we show an example of set **Trunk** so defined. The set **Trunk** may contain small branch components and parts of ground, but most of these can be easily removed by determining the connected components of **Trunk** and then removing all the small components. Furthermore, using the largest component of **Trunk**, the axis of the trunk can be estimated and **Trunk** can be redefined as those large components whose mean is less than, say, half a meter away from the axis.

Next, define the base **TBase** of the trunk as the lowest part (and its neighbors) of the lowest component of **Trunk** (e.g., a 20-cm slice from the lowest point). Most of the ground and other sets that are not part of the tree can now be removed easily. First, define the set **Ground** containing the cover sets connected to the tree base **TBase**, but not connected to it through **Trunk**. The set **Ground** now includes ground and vegetation points. Next, define the tree component starting from **TBase** by expanding **TBase** as long as possible, but prevent the expansion into **Ground**. This is now the first tree component and **TBase** is its base (see the middle frame of Figure 3).

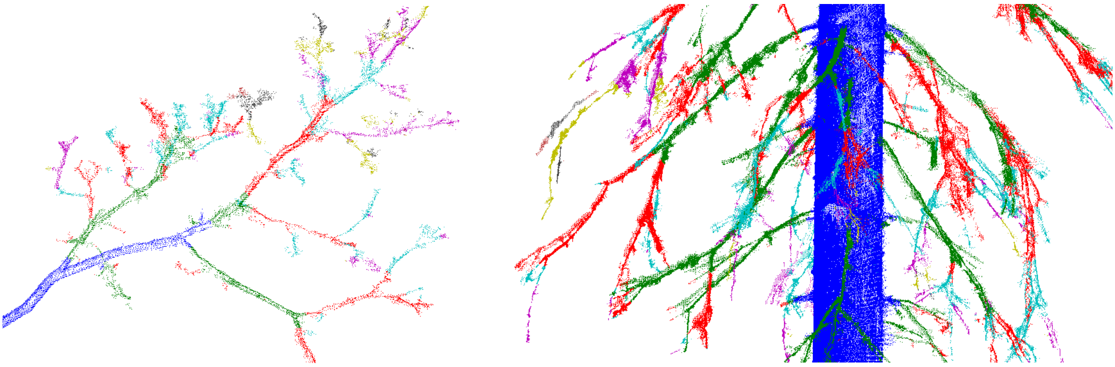
Next, determine the connected components of the point cloud from which **Ground** and the first tree component are removed. Then find out which of these components are part of the tree. This can be done heuristically by defining the means of the components and then checking their heights from the ground and distances to the trunk axis. The final classification of the connected components is shown in the last frame of Figure 3.

The bases of the other tree components need to be determined for the segmentation process. There is no easy and fast way to do this with full reliability, and for some components, the defined base may not be the right one. A wrong base can mix up the branching-relation, *i.e.*, which segment is the *parent* branch and which is the *child* branch. However, our heuristic apparently works most of the time. If the component shares common sets with **Trunk**, then we select the lowest common set as its base, and this base defines a trunk segment. For other components, we first project the component into its largest principal component to find its two ends in the principal direction. Then, we select the end that is closer to the trunk axis as the base of the component.

2.8. Segmentation

When the tree components and their bases are determined, the next step is to segment these components into branches. Each component is partitioned into segments that correspond to the whole or part of a real branch or trunk. In particular, segments should not have any bifurcations. This kind of segmenting also defines the tree structure, *i.e.*, the branching-relations of the *child* and *parent* branches for each branch. It is also straightforward to fit cylinders to these segments. Examples of segmented tree parts are shown in Figure 7 and Figure 1 shows a segmented tree.

Figure 7. Examples of segmented tree parts. (**Left**) A segmented branch originating from the trunk of an maple. (**Right**) Close-up of a segmented Norway spruce.



2.8.1. Overview of the Algorithm

The segmentation process takes place at the level of the cover sets, and the basic tool is their neighbor-relation. The procedure starts from the base of a tree component and, step by step, a *cut region* and its extension, a *study region*, move along the component (see Figure 8). The cover sets are the building bricks of the tree surface and the cut region is a layer of these bricks that separates the component into two parts. At each step the cut region moves to the neighboring layer of cover sets. Every cut region is expanded or "grown" along the tree component into the study region that consists of multiple layers of cover sets. Each study region is checked for connectivity to find out possible bifurcations. If the study region is not connected, then it is further checked if its separate parts are the beginnings of new branches or parts of the current segment. The part of the cut region in a branch becomes a base for a new segment to be determined later. The parts of the cut region that are not new segment bases are added to the segment, and this way the segment grows one layer a time. The segment is expanded until no cut regions can be constructed or none of the separate parts of the study region are clear continuations of the segment. A schematic picture of this process is shown in Figure 8.

Figure 8. A schematic picture of the bifurcation recognition process. The cut region (red) and its extension, the study region (light red), move along the tree component (brown) and construct the segment (green). When the cut and study regions move ahead through a bifurcation, such as a branch, the regions will no longer be connected, and the part of the cut region belonging to the branch becomes a new branch base.

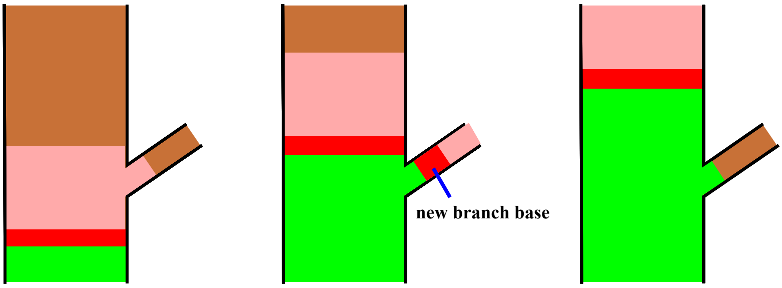
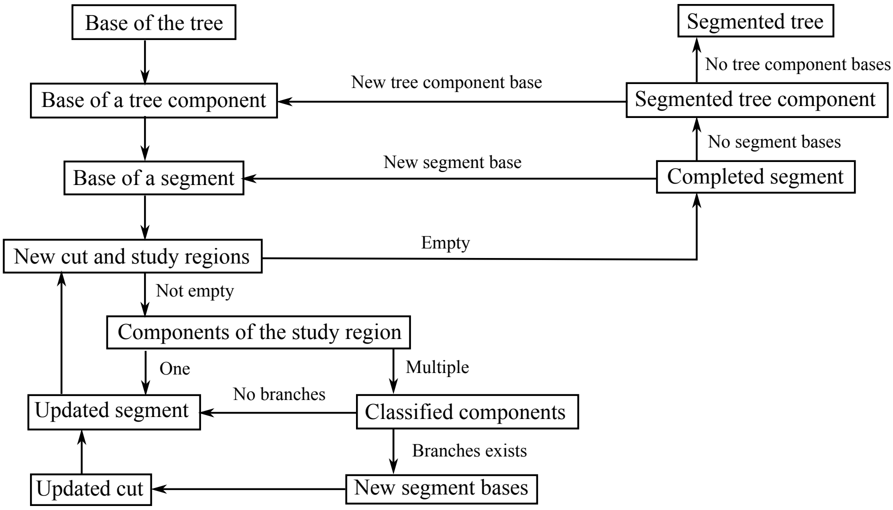


Figure 9. The segmenting process.



Notice that each new segment base found also defines the tree structure step by step. Furthermore, because only neighbor-relations and relative sizes are needed, the process is scale independent. However, the accuracy and density of the scanned data points and the size of the cover sets define a limit for the separation: If the accuracy of the scanned points is of the order of the width of the gap between neighboring parallel branches, then the two branches are merged into one in the data. Also, if the density of the points is so small that the distance between the closest points is about the same as the gap between branches, then the gap and thus the branches are indistinguishable. Finally, if the branches are so small that they are contained (in the transversal direction) in a single cover set, they are also indistinguishable, because segments are collections of cover sets. Figure 9 shows the segmentation process as a flow chart. Details of the segmentation algorithm are given in the Appendix.

2.8.2. Remarks

Because each segment is constructed one layer of cover sets at a time, these layers partition the segments. These layers should be saved so that they can be used in the cylinder reconstruction, where the segments are first divided into smaller regions. Furthermore, especially in the trunk where the cover sets are small compared to the segment size, gaps in the data due to self-shadowing may lead to wrong segments. These gaps may lead to small segments along the trunk which are classified as child segments of the trunk segment that continues past them, *i.e.*, the child and parent segments are part of the same real trunk at the same height of the trunk. When cylinders are fitted to these small segments, the cylinders are strongly overlapping and nearly parallel to the cylinders in the parent trunk segment. This can be corrected by comparing the cylinders in these segments and removing the overlapping ones in the child segments.

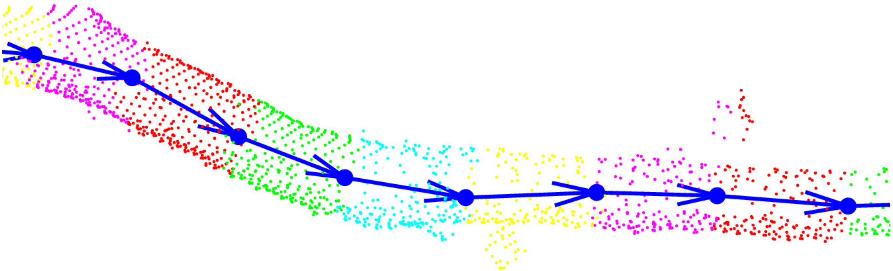
2.9. Cylinder Reconstruction

Each segment is reconstructed with successive cylinders locally approximating the radius and orientation of the segment (branch). In the process, also the succession relations of the fitted cylinders are recorded so that the tree structure can be expressed in terms of the child/parent relation of the cylinders.

The cylinders are expressed parametrically with seven real numbers (the radius and the components of the axis direction vector and the axis position vector) and the parameters are optimized using the total least squares method [37]. This nonlinear optimization problem requires iterative solution methods with good initial estimates for the parameter values [38]. Another possible way to fit cylinders is to fix the axis direction and project the points into a plane orthogonal to the axis. Then we fit a circle to the projected points in the least square sense [39].

Our adopted approach starts from the base of a segment and then subdivides it into small successive pieces. These pieces consist of layers of cover sets and, here, the partition of the segment into layers in the segmentation process can be used. The number of layers should be such that the length of the pieces is at least equal to the estimated diameter of the segment. The center points of the successive pieces define vectors that define the *regions*, which are later approximated with cylinders. The regions form a new subdivision of the segment and each region consists of those points of two consecutive pieces whose projected points are between the start and end points of the vector joining the center points of the pieces (see Figure 10). These vectors and their starting points are also used as the initial estimates for the axis direction and starting point in the cylinder fitting. The initial estimate for the radius can be the mean or median distance of the points in the region from its estimated axis. Longer regions are more tolerant of noise, outliers, and imperfect point cover, but in some cases, they may not follow the radius and direction of the branch as accurately as shorter regions. The effects of outliers and noise can also be reduced by a second fitting: the cylinder of the first fitting is used to identify points far away from the cylinder that can be removed from the point set of the second fitting.

Figure 10. Regions for cylinder fitting. Different colors denote the regions defined by the blue vectors.



When all the regions of a segment are approximated by an adjusted sequence of cylinders, the radii of the cylinders are checked using prior knowledge: the diameter of a branch is practically always smaller or only a little larger than the diameter of its parent, and the diameter of a branch usually decreases away from its base. Moreover, to make the sequence more continuous (e.g., to close small gaps between the cylinders) and to correct for the effects of radius changes, the starting points and the axes are modified when needed.

2.10. Completing the Cylinder Model

When all the segments are reconstructed with cylinders, the cylinder model may still be refined. The point cloud may have multiple connected components, in which case there are gaps between these components. There may also be clear gaps between a parent cylinder and its child cylinder which is not an extension. Thus there may be gaps in the cylinder model which can produce errors for the tree statistics. To reduce these errors, we identify small gaps between the fitted cylinders and then fit cylinders to these gaps using only the previously fitted cylinders as data.

There are two basic cases. In the first case, there are two nearby and nearly parallel cylinders, one without a parent (C) and the other without an extension (A) (see Figure 11). Define a vector from the top of A to the bottom of C. Then this vector defines the axis of the new cylinder B whose radius is the mean of the radii of A and C. Moreover, B not only joins the cylinder chains but also the underlying segments since the segments of A and C are joined into one single segment. In this way, the tree structure is also refined.

In the second case, there is a cylinder C without a parent, and, close to it, there is a second cylinder A which is transversal to C and has larger radius (see Figure 12). The idea is to add a new cylinder B as a child of A such that the radius of B equals that of C. Furthermore, the axis line of B will intersect the axis of A, and the axes of C and B are adjusted to be as parallel as possible. Again, the added cylinder refines the tree structure.

Figure 11. Filling gaps. The green cylinder (A) has no extension, and the nearby blue cylinder (C) has no parent. The gap between these two cylinders is filled by the red cylinder (B).

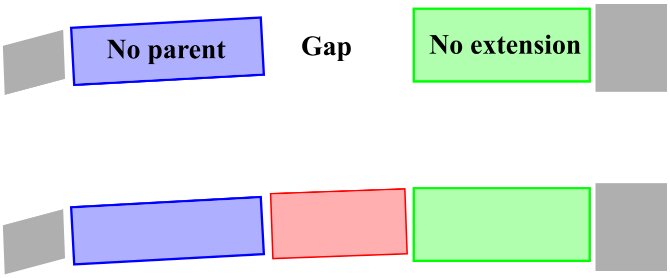
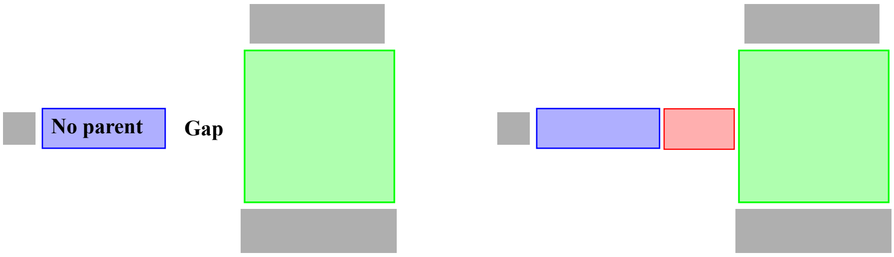


Figure 12. Filling gaps. The blue cylinder (A) has no parent, but there is a nearby green cylinder C with a larger radius. The gap between these two cylinders is filled by the red cylinder (B).



3. Testing and Results

In this section we first define different tree metrics that can be approximated from completed tree models. Then we test our method using first simulated TLS data from artificial tree models. Next, we compare model results for caliper measurements of small branches of a real tree. Finally, we present complete models and results for real trees.

3.1. Tree Metrics

A large variety of (external) statistical measures and other characteristics of the tree can be approximated from the cylinder model. These are, e.g., the total volume of the whole tree or its parts such as the part of trunk with certain radii and the branches of certain size. The total volume and length of the trunk and branches can be also estimated, but depending on the quality of data (e.g., the extent of surface cover), the upper parts of the tree may be quite incomplete. Similarly, the sum of the lengths of the cylinders gives a good estimate of the total branch length in the tree. Trunk and branch profiles (taper functions) can also be easily estimated.

There are other statistical data that can be approximated from the cylinder model, e.g., the number of sub-branches originating from branches, the angles between child and parent branches, and the averages of these. An important concept is the branch size distribution, which characterizes a tree in many ways. From the cylinder models we determine the distribution as follows: we assign the branch cylinders according to their diameter into bins (under 1 cm, over 1 cm but under 2 cm, over 2 cm but under 3 cm, *etc*.). and then compute how much of the total branch volume there is in each bin.

3.2. Testing with Artificial Trees

The basic performance of the procedure for the parts existing in the data was validated in [40], where random but uniformly distributed samples were taken from a simple artificial tree. Random samples mimicking typical target sizes estimated the error level to remain within the expect bounds; *i.e.*, within roughly 5–10% for the characteristics of the whole part existing in the data, depending on the reconstructed quantity.

To test the method with more realistic data and the dependency of some results on the main input parameters, the cover parameters d and r, we have used simulated TLS scanning from a complex artificial 3D tree model (see Section 2.1). First, we fixed the cover set diameter d to 2.0 cm and checked how changes in the radius r, which controls the geometric characteristics and neighbor-relation, affect reconstructions. The resulting partition C_P was the same for each case. The results for the radius test are shown in Table 1. Figure 1 shows the segmented point cloud and the final cylinder model when $d = 2.0$ cm and $r = 2.4$ cm.

Table 1. Effects of the cover set radius. #comps, #branch, and #1st branch columns give the number of tree components, branches reconstructed with cylinders, and first-order branches, respectively. The time column gives the time for completing the model (MacBook Pro, 2.8 GHz, 8 GB).

Radius	**Tot. vol.**	**Trunk**	**Branch**	**Trunk**	**Branch**	**#comps**	**#branch**	**#1st**	**Time**
(cm)	(dm^3)	**vol.** (dm^3)	**vol.** (dm^3)	**len. (m)**	**len. (m)**			branch	(s)
orig	665	348	316	17.2	2,665	1	13,102	99	
2.0	568	348	220	17.2	1,970	534	7,520	95	430
2.2	566	348	218	17.2	1,993	357	7,448	96	435
2.4	581	348	233	17.2	1,985	258	7,320	99	430
2.6	593	348	245	17.2	1,960	186	7,128	103	430
2.8	573	320	253	10.5	1,933	150	6,865	52	445
3.0	579	327	252	11.1	1,885	117	6,624	56	430

Obviously, as the radius increases, the connectivity increases as well, which is reflected in the decreasing number of tree components. We also see that the total branch length and the number of branches are little affected by the increasing radius, but they decrease somewhat. This is again expected because of the increasing connectivity (larger neighborhoods). Because there are about 1,778 m of branches with a diameter under 1 cm in the original tree, it is clear that not all of them are sampled sufficiently by the point cloud. This explains the difference between the total branch lengths of the original and reconstructed. Nearly all or all the first-order branches (branches originating from the trunk) are found, and with a larger radius also some second-order branches very close to trunk are classified as first-order branches. The evaluation of the branching structures beyond the first order is complicated because there are lots of separate components and the order of the "base branch," even if that exists, is unknown. However, visual inspection verifies that the branching structure is well defined inside the components in the sense of branch separation. We also see that, for a large radius, the trunk classification, due to, e.g., the segmentation and initial trunk component classification, varies quite a lot in the sense of trunk length. However, most of the trunk volume is still covered as the length differences pertain to thin parts. From the data we conclude that in this case the suitable radius r is about 2.0–2.4 cm, and thus the radius can be a little larger than the diameter d.

Next we used multiple different sizes for cover sets to see the effect of the diameter d on the results (in each case the radius is 0.4 cm larger than the diameter). Figure 13 shows that the trunk profiles of the original and reconstructed trees are little affected by the size of the cover set, except for small variations in the upper part for the smallest cover sets (see also the trunk volumes in Table 2). This is expected because most of the trunk is visible in the data and it is covered quite well all around. Figure 14 shows the branch size distribution of the original and reconstructed trees. First, only part of the smallest branches (diameter under 2 cm) exists in the data and thus only part of their volume is reconstructed. Second, branches whose diameters are much smaller than the cover set usually cannot be sharply segmented. Then a large cover set may contain points from multiple nearby small branches or from a branch and its child branches. Thus we see that the larger the cover sets, the larger the bins in the volume distributions and the less the number of branches and components. This can be seen also in Table 2: the total volume and length of the branch cylinders increase and decrease, respectively, as d increases. When the trunk is correctly defined, then all (or nearly all) first-order branches are found, and with larger d few second-order branches close to trunk are classified as first-order branches. Again, the branching structure is well defined for most of the tree, and the differences are in the smallest branches.

Figure 13. Trunk profiles.

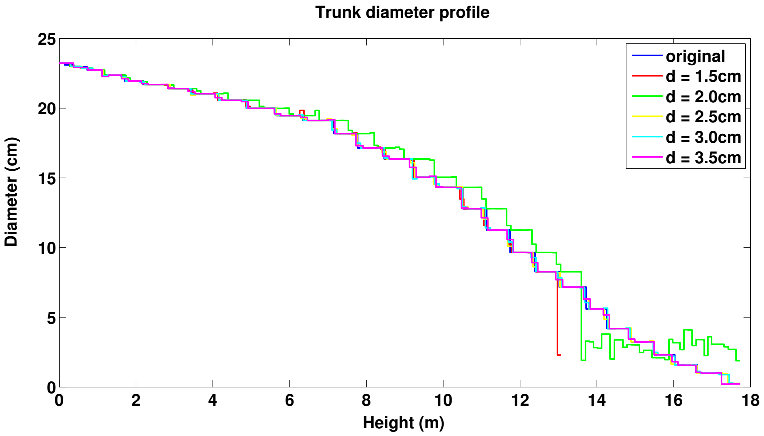
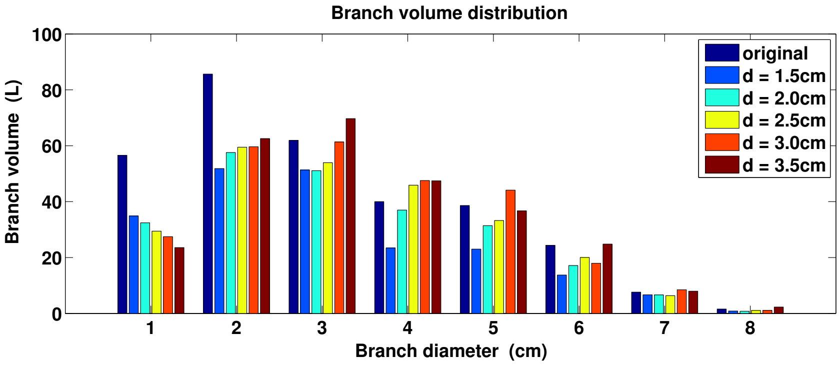


Table 2. Effects of the cover set diameter, with columns as above.

Diam	**Tot. vol.**	**Trunk**	**Branch**	**Trunk**	**Branch**	**#comps**	**#branch**	**#1st**	**Time**
(cm)	(dm^3)	**vol.** (dm^3)	**vol.** (dm^3)	**len. (m)**	**len. (m)**			**branch**	**(s)**
orig	665	348	316	17.2	2,665	1	13,102	99	
1.5	548	342	206	13.1	2,060	393	8,385	62	720
2.0	582	348	234	17.2	1,985	258	7,315	98	480
2.5	598	348	250	17.2	1,825	164	6,133	101	375
3.0	632	348	284	17.2	1,652	119	5,076	103	315
3.5	623	348	275	17.2	1,467	92	4,059	97	270

Figure 14. Branch size distribution.

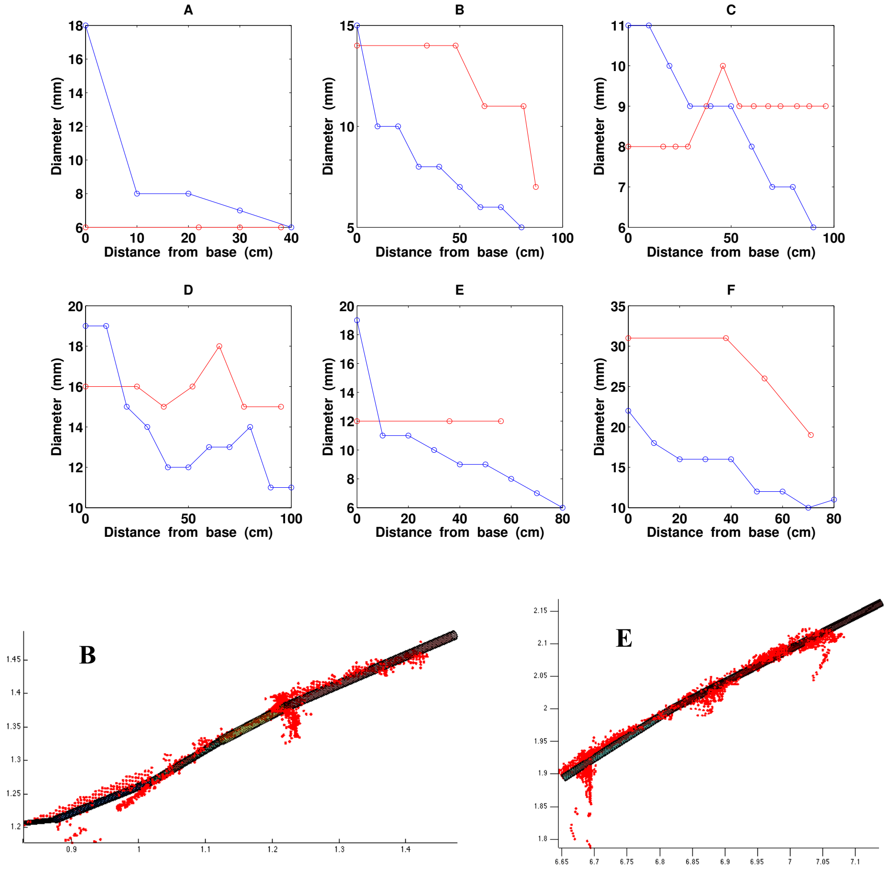


3.3. Testing with Real Measurements for Small Branches

There are many sources of errors in the scanning, particularly for small branches. The size of the laser spot is about 5 mm when it hits the branch (the spot size depends on the distance). There is also a small error when scans from different directions are registered into a common coordinate system (some millimeters). Windy conditions can add some error due to movement of the branches. Furthermore, the scanning beam can be reflected from parts outside the actual surface. For example, there is a certain amount of data noise in coniferous trees throughout the year due to needles. Thus, it is clear that the smallest branches (diameter under 3 cm) cannot be scanned very accurately. In order to check if the cylinder reconstruction works at all for branches whose diameter is near the scanning accuracy, we have measured some branch profiles for the spruce shown in Figure 2 using caliper. The branches have small diameters (about 1–3 cm) and their height above the ground is some 1–3 m. In Figure 15, we show some caliper measurement results and compare them with the results obtained from the cylinder model.

Assuming that the caliper measurements are accurate, the results show that the diameter measurement error for very small branches is of the order of one centimeter, and usually the computed diameter is larger than the measured one. There may, of course, be different biases in the two measurements (caliper and cylinders) which affect the results. For example, the cross-section of small branches is rarely circular. However, the accuracy of the cylinder fitting for branches whose diameter is near the scanning accuracy is as expected. Moreover, the orientations and locations of branches and the tree structure can still usually be reliably reconstructed for small branches, as can be seen from the bottom frames of Figure 15.

Figure 15. Comparison of measured and computed branch profiles for thin branches. The graphs below show the measured (blue) and computed (red) profiles: diameter in mm as a function of branch length in cm. Below are the fitted cylinders for the profiles B and E. The red points are the laser data points.



3.4. Complete Models from Real TLS Data

We have scanned four trees, a Norway spruce and maple, and two Scots pines, the first one shown in Figure 3. The trees have been scanned from three different directions to have a comprehensive cover of the branching structure (see the scanner specifications in Section 2.1). Complete cylinder models for a Norway spruce and maple (without leaves) are shown in Figure 2. In Table 3, we show some statistics computed for the trees. Branch size distributions for the trees are shown in Figure 16. Notice that because the trunks of the maple and the scots pine 2 bifurcate, a significant proportion of the branch volume is in large branches with diameter over 10 cm.

Figure 16. Branch size distributions.

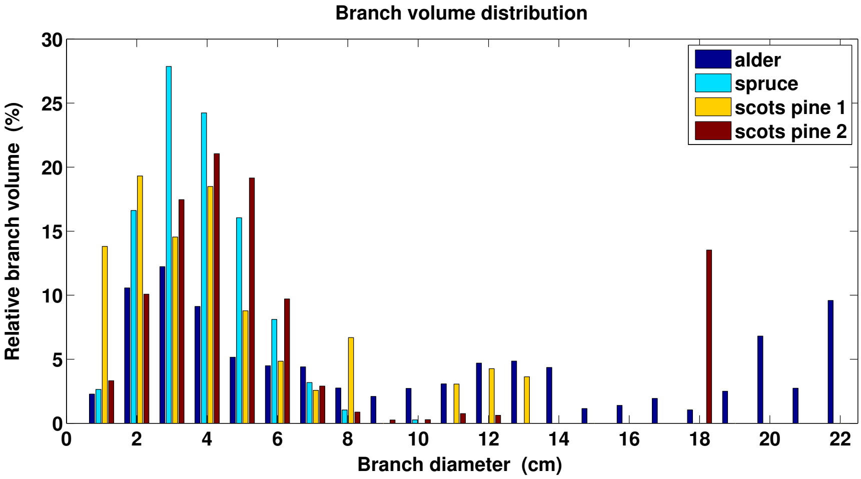


Table 3. Tree measures computed from the cylinder models. The time column gives the time for model completion (MacBook Pro, 2.8 GHz, 8 GB). The cover parameters were $d = 2.0$ cm and $r = 2.5$ cm.

Tree	Tot. Vol.	Trunk Vol.	Branch Vol.	Trunk Height	Tot. Height	Branch Len.	Time
	(dm^3)	(dm^3)	(dm^3)	(m)	(m)	(m)	(s)
spruce	1,090	790	300	14.7	22.0	720	1,160
scots pine 1	410	370	40	13.4	20.5	150	245
scots pine 2	670	450	220	9.4	16.4	380	740
maple	690	170	520	7.9	11.2	720	760

4. Discussion

The results from Section 3 demonstrate that our method is able to automatically build models that reconstruct the parts existing in the data. With near-perfect data, such as the simulated data, where potential error sources (registration, wind, *etc.*) can be controlled and eliminated, the method is able to reconstruct the tree accurately. Even small branches near the accuracy level of scanning can be reconstructed, and their branching structure is retained in the model. Overall, the results seem quite good and, e.g., trunk volumes are very close to the real ones. Compared to a recent voxel-based method [41], where there was much better cover of the tree (4–5 scanning positions and 20–60 million points) and smaller unit size (1 cm), the errors in the total volume are about the same. In future studies, we plan to conduct more validation tests with large field measurements.

Most of the parts missing in the data are not reconstructed, and this affects different tree metrics estimated from the model. In particular, the top parts of (high) trees are usually poorly covered by TLS data (see Figure 2). However, because the lower parts have a better cover, it should be possible to use the modeled lower parts to give accurate corrections to the total volumes and branch size distributions in the upper parts. This is a subject for further study.

The method does not model the foliage or needles, and if these are present, they can add noise, make nearby branches indistinguishable, prevent the visibility of many parts, *etc*. Needles, especially, can make the smallest branches appear larger in diameter then they are. Again, however, these could be taken into account with proper calibration, but this requires extensive field measurements and testing.

The segmentation works well and the separation of the branches comparable or larger in sizes to the cover sets is nearly perfect. Large noise, if it cannot be properly removed, can affect the separation of branches but this cannot be helped. Thus individual branches can be analyzed and their topological relation to the other branches can be determined. However, gaps in the data breaks the data into separate components and the relations of the components is harder to determine accurately. The gap-filling procedure presented in Section 2.10 was realized so that only very clear cases were allowed. Thus, only a few tens of gaps were filled in any of the reconstructed models. This procedure will be developed further.

As shown in Section 3.2, the method is quite robust against the selection of the cover parameters d and r. This is especially true for parts that are large enough compared to d; *i.e.*, when the sampling size is small enough, the reconstruction works well. However, this method and all methods reconstructing the whole tree structure from point clouds are quite sensitive to the quality of the measurements. Therefore, for example, if the point density is too low or there is lot of movement in the tree during the measurements, the smallest branches cannot be reliably reconstructed.

We have implemented the method with MATLAB, and the main inputs are the point cloud as well as the cover and filtering parameters. Other parameters determine, e.g., if a component is part of the tree, whose parts are initially classified as trunk, and converge criteria for cylinder fitting. If these parameters are approximately tuned, the method is automatic and no human intervention is needed. The computation time and memory required mainly depend on the number of cover sets used, and for most of the steps, the time depends linearly on the number of sets. Especially for the segmentation, which is the most time-consuming step, the required time is roughly proportional to the total length of the branches. However, the time and hardware requirements are modest; the whole process (with data sets of 1–5 million points) takes from minutes to half an hour (see the tables in Section 3) with a MacBook Pro (2.8 GHz, 8 GB).

In future work, we will apply our method to a larger number of trees to draw some statistical inferences, and we will analyze exposed stumps and roots of trees, for which there is so far very little size and structure data (Liski *et al.*, in preparation). We will also compare the results obtained with conical-elliptic shapes and deformable cylinders to estimate the inherent systematic errors of the procedure in some attributes extracted. Furthermore, the computational method and its realizations still need some testing, validations, and further developments, such as adaptive parameter tuning, so that a general software package can be compiled for a variety of end-users.

5. Conclusions

We have presented the main steps and demonstrated the potential and operability of a method for constructing automatically comprehensive precision models of trees from TLS point clouds. The method is fast, producing almost immediately a model that is compact compared to the measurement data. In the model, the diameters and orientations of the trunk and branches are locally approximated with cylinders. The branching structure, *i.e.*, the parent–child relation of the branches, is also recorded in the model. The method is independent of the absolute scale and can thus be used to analyze the tree quantitatively down to the level of detail allowed by the data.

The cover-set approach is the driving engine behind our method. It renders the point cloud as a collection of topologically, geometrically, and hierarchically defined units that can be analyzed with standard fitting procedures to obtain orientations, sizes, *etc*. The other key principle presented here is the wholesale processing of the point cloud to obtain a simple easily accessible model that is designed to contain most of the necessary trunk–branch information even without defining that information *a priori*.

Using the model, we can carry out quantitative analysis of the structure and size properties of the tree. For example, the total volume of the whole tree or any of its parts, such as the trunk or branches, can be approximated from the model. The trunk and branch profiles, or statistical distributions such as the branching angle distribution, can be determined from the model. In particular, the model yields the branch size distributions, which can be used to improve tree quality and forest carbon cycle estimations.

The reference measurements from small real tree branches show that the method is able to reconstruct the branches with reasonable accuracy (error under 1 cm). The results from the modeled TLS data show that the volume of trunks and large branches can be reconstructed accurately (few percent error) even when large portions of tree parts are not sampled by measurements. The data need not be as comprehensive and dense as required for voxel-based methods.

In the future, we will validate the method further with a large number of trunk and branch measurements from real trees. We will also develop the method further, e.g., by utilizing generalized cylinder shapes.

Together with the computational method presented, laser scanning provides a fast and efficient means to collect essential geometric and topological data from trees, thereby substantially increasing the available data from trees.

Acknowledgements

This study was financially supported by the Academy of Finland projects: *Modelling and applications of stochastic and regular surfaces in inverse problems* and *New techniques in active remote sensing: hyperspectral laser in environmental change detection*, and the Finnish Centre of Excellence in Inverse Problems Research. We thank Jari Liski and Risto Sievänen for valuable discussions and suggestions.

References

1. Peuhkurinen, J.; Maltamo, M.; Malinen, M.; Pitkänen, J.; Packalén. P. Preharvest measurement of marked stands using airborne laser scanning. *For. Sci.* **2007**, *53*, 653–661.
2. Repola, J. Biomass equations for Scots pine and Norway spruce in Finland. *Silva Fennica* **2009**, *43*, 625–647.
3. Van Leeuwen, M.; Nieuwenhuis, M. Retrieval of forest structural parameters using lidar remote sensing. *Eur. J. For. Res.* **2010**, *129*, 749–770.

4. Dassot, M.; Constant, T.; Fournier, M. The use of terrestrial LiDAR technology in forest science: Application fields, benefits and challenges. *Ann. For. Sci.* **2011**, *68*, 959–974.
5. Hyyppä, J.; Jaakkola, A.; Hyyppä, H.; Kaartinen, H.; Kukko, A.; Holopainen, M.; Zhu, L.; Vastaranta, M.; Kaasalainen, S.; Krooks, A.; *et al.* Map Updating and Change Detection Using Vehicle-Based Laser Scanning. In *Proceedings of 2009 Urban Remote Sensing Joint Event*, Shanghai, China, 20–22 May 2009.
6. Holopainen, M.; Vastaranta, M.; Kankare, M.; Räty, M; Vaaja, M.; Liang, X.; Yu, X.; Hyyppä, J.; Hyyppä, H.; Viitala, R.; *et al.* Biomass estimation of individual trees using stem and crown diameter TLS measurements. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2011**, *38*, 91–95.
7. Henning, J.G.; Radtke, P.J. Detailed stem measurements of standing trees from ground-based scanning lidar. *For. Sci.* **2006**, *52*, 67–80.
8. Hopkinson, C.; Chasmer, L.; Young-Pow, C.; Treitz, P. Assessing forest metrics with a ground-based scanning lidar. *Can. J. For. Res.* **2004**, *34*, 573–583.
9. Liang, X.; Litkey, P.; Hyyppä, J.; Kaartinen, H.; Vastaranta, M.; Holopainen, M. Automatic stem mapping using single-scan terrestrial laser scanning. *IEEE Trans. Geosci. Remote Sens.* **2012**, *50*, 661–670.
10. Pfeifer, N.; Gorte, B.; Winterhalder, D. Automatic reconstruction of single trees from terrestrial laser scanner data. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2004**, *35(B5)*, 114–119.
11. Thies, M.; Pfeifer, N.; Winterhalder, D.; Gorte, B.G.H. Three-dimensional reconstruction of stems for assessment of taper, sweep and lean based on laser scanning of standing trees. *Scand. J. For. Res.* **2004**, *19*, 571–581.
12. Watt, P.J.; Donoghue D.N.M. Measuring forest structure with terrestrial laser scanning. *Int. J. Remote Sens.* **2005**, *26*, 1437–1446.
13. Moorthy, I.; Miller, J.R.; Hu, B.; Chen, J.; Li, Q. Retrieving crown leaf area index from an individual tree using ground-based lidar data. *Can. J. Remote Sens.* **2008**, *34*, 320–332.
14. Pfeifer, N.; Winterhalder, D. Modelling of tree cross sections from terrestrial laser-scanning data with free-form curves. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2004**, *36*, 76–81.
15. Binney, J.; Sukhatme, G.S. 3D Tree Reconstruction from Laser Range Data. In *Proceedings of ICRA '09 IEEE International Conference on Robotics and Automation*, Kobe, Japan, 12–17 May 2009; pp. 1321–1326.
16. Bucksch, A.; Fleck, S. Automated detection of branches dimensions in woody skeletons of fruit tree canopies. *Photogramm. Eng. Remote Sensing* **2011**, *77*, 229–240.
17. Côté, J.-F.; Widlowski, J.-L.; Fournier, R.A.; Verstraete, M.M. The structural and radiative consistency of three-dimensional tree reconstructions from terrestrial lidar. *Remote Sens. Environ.* **2009**, *113*, 1067–1081.
18. Filin, S.; Pfeifer, N. Neighborhood systems for airborne laser data. *Photogramm. Eng. Remote Sensing* **2005**, *71*, 743–755.
19. Maas, H.-G.; Bienert, A.; Scheller, S.; Keane, E. Automatic forest inventory parameter determination from terrestrial laser scanner data. *Int. J. Remote Sens.* **2008**, *29*, 1579–1593.

20. Rutzinger, M.; Pratihast, A.K.; Oude Elberink, S.; Vosselman, G. Detection and modeling of 3D trees from mobile laser scanning data. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2010**, *38*, 520–525.
21. Gorte, B.G.H.; Pfeifer, N. Structuring laser scanned trees using 3D mathematical morphology. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2004**, *35*, 929–933.
22. Bucksch, A.; Lindenbergh, R. CAMPINO—A skeletonization method for point cloud processing. *ISPRS J. Photogramm.* **2008**, *63*, 115–127.
23. Bucksch, A.; Lindenbergh, R.; Menenti, M. SkelTre-Robust skeleton extraction from imperfect point clouds. *Vis. Comput.* **2010**, *26*, 1283–1300.
24. Verroust, A.; Lazarus, F. Extracting skeletal curves from 3D scattered data. *Vis. Comput.* **2000**, *16*, 15–25.
25. Raumonen, P.; Kaasalainen, S.; Kaasalainen, M.; Kaartinen, H. Approximation of volume and branch size distribution of trees from laser scanner data. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2011**, *38(5/W12)*, 79–84.
26. Åkerblom, M.; Raumonen, P.; Kaasalainen, M.; Kaasalainen, S.; Kaartinen, H. Comprehensive Quantitative Tree Models from TLS Data. In *Proceedings of 2012 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Munich, Germany, 22–27 July 2012; pp. 6507–6510.
27. Kaasalainen, S.; Jaakkola, A.; Kaasalainen, M.; Krooks, A.; Kukko, A. Analysis of incidence angle and distance effects on terrestrial laser scanner intensity: Search for correction methods. *Remote Sens.* **2011**, *3*, 2207–2221.
28. Disney, M.I.; Lewis, P.; Saich, P. 3D modelling of forest canopy structure for remote sensing simulations in the optical and microwave domains. *Remote Sens. Environ.* **2006**, *100*, 114–132.
29. Leersnijder, R.P. *PINOGRAM: A Pine Growth Area Model*; WAU Dissertation 1499; Wageningen Agricultural University: Wageningen, The Netherlands, 1992.
30. Disney, M.I.; Lewis, P.; Gomez-Dans, J.; Roy, D.; Wooster, M.; Lajas, D. 3D radiative transfer modelling of fire impacts on a two-layer savanna system. *Remote Sens. Environ.* **2011**, *115*, 1866–1881.
31. Widlowski, J.-L.; Robustelli, M.; Disney, M.I.; Gastellu-Etchegorry, J.-P.; Lavergne, T.; Lewis, P.; North, P.R.J.; Pinty, B.; Thompson, R.; Verstraete, M.M. The RAMI Online Model Checker (ROMC): A web-based benchmarking facility for canopy reflectance models. *Remote Sens. Environ.* **2008**, *112*, 1144–1150.
32. Hancock, S.; Lewis, P.; Foster, M.; Disney, M.I.; Muller, J.-P. Measuring tree height over topography and understory vegetation with dual wavelength lidar: A simulation study. *Agric. For. Meteorol.* **2012**, *161*, 123–133.
33. Bae, K.-H.; Lichti, D.D. A method for automated registration of unorganised point clouds. *ISPRS J. Photogramm.* **2008**, *63*, 36–54.
34. Mitra, N.J.; Nguyen, A.; Guibas, L. Estimating surface normals in noisy point cloud data. *Int. J. Comput. Geometry Appl.* **2005**, *14*, 261–276.
35. Jolliffe, I.T. *Principal Component Analysis*, 2nd ed.; Springer-Verlag: New York, NY, USA, 2002.
36. Demantké, J.; Mallet, C.; David, N.; Vallet, B. Dimensionality based scale selection in 3D lidar point clouds. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2011**, *38(5/W12)*, 97–102.

37. Lukacs, G.; Martin, R.; Marshall, D. Faithful Least-Squares Fitting of Spheres, Cylinders, Cones and Tori for Reliable Segmentation. In *Proceedings of the 5th European Conference on Computer Vision*, Freiburg, Germany, 2–6 June 1998; pp. 671–686.
38. Madsen, K.; Nielsen, H.B.; Tingleff, O. *Methods for Non-Linear Least Squares Problems*, 2nd ed.; Informatics and Mathematical Modelling, Technical University of Denmark: Lyngby, Denmark, 2004.
39. Gander, W.; Golub, G.H.; Strebel, R. *Fitting of Circles and Ellipses—Least Squares Solution*. Available online: ftp.inf.ethz.ch as doc/tech-reports/1994/217.ps (accessed on 21 January 2013).
40. Åkerblom, M. Quantitative Tree Modeling from Laser Scanning Data. M.S. Thesis, Tampere University of Technology, Tampere, Finland, 2012.
41. Vonderach, C.; Voegtle, T.; Adler, P. Voxel-base approach for estimating urban tree volume from terrestrial laser scanning data. *Int. Arch. Photogr. Remote Sens. Spat. Inf. Sci.* **2012**, *39-B8*, 451–456.

Appendix: Details of the Segmentation Algorithm

We define variables that are sets containing cover sets:
TBase: the base of the current tree component
Base: the base of the current segment
Seg: the current segment
Cut: the cut region
PreCut: the previous cut region
Study: the study region
Forb: the sets to which all future segments are forbidden to expand
ForbSeg: the sets to which the current segment is forbidden to expand

Let **S** be a set of cover sets. Then $Ext(\mathbf{S})$ is the set of cover sets containing the cover sets in **S** and all the neighboring cover sets of **S**. The number of cover sets in **S** is denoted by $\#\mathbf{S}$.

Initially **Forb** contains the cover sets that are not part of the tree and **TBase** is the base of the trunk. These are defined in Section 2.7. Then the segmentation of the point cloud is done with the following algorithm.

A1. Initial Stage of Each Tree Component

Set the variables as follows:

$\mathbf{Seg} = \mathbf{TBase}$
$\mathbf{PreCut} = \mathbf{TBase}$
$\mathbf{Forb} = \mathbf{Forb} \cup \mathbf{TBase}$
$\mathbf{ForbSeg} = \mathbf{Forb}.$

A2. Construct the Cut Region and Study Regions

Define **Cut** by expanding **PreCut** into the tree component:

$$\mathbf{Cut} = Ext(\mathbf{PreCut}) \setminus \mathbf{ForbSeg}$$

If **Cut** is empty, go to step 6. Otherwise, if at the beginning of the segment, determine the number N of cover set layers in the study regions of this segment. For this, estimate the diameter of the segment, and require that the study regions should be about as high as the estimated diameter, but at least $N = 2$. Then define **Study** by expanding **Cut** into the tree component:

$$\text{set } \mathbf{Study} = \mathbf{Cut} \text{ and iterate } (N - 1)\text{−times } \mathbf{Study} = Ext(\mathbf{Study}) \setminus \mathbf{ForbSeg}.$$

A3. Determine the Connected Components of Study

If the cut region is not expanded much, *i.e.*, if e.g., $\#\mathbf{Study} < 2\#\mathbf{Cut}$ holds, set the number of connected components to one. Otherwise, determine the connected components $\{\mathbf{C}_i\}$ (see Section 2.4). If the number of components is one, proceed to step 6, otherwise to step 4.

A4. Classify the Components of Study

Each component $\mathbf{C}_i$ is checked if it is a new branch or part of the current segment, e.g., a continuation of the segment. The base $\mathbf{B}_i$ of a component $\mathbf{C}_i$ is the part of the component belonging to the cut region, *i.e.*, $\mathbf{B}_i = \mathbf{C}_i \cap \mathbf{Cut}$. Then classify $\mathbf{C}_i$ as part of the current segment if (1) component is its base, *i.e.*, $\mathbf{C}_i = \mathbf{B}_i$ holds, or (2) if the base is very small compared to cut region and the expansion of the component is very small; if, e.g., $\#\mathbf{B}_i < 0.05\#\mathbf{Cut}$ and $\#\mathbf{B}_i > 0.5\#\mathbf{C}_i$ hold, or (3) if there are only a few points in $\mathbf{C}_i$. Furthermore, if the base is nearly all of the cut region (say, $\#\mathbf{B}_i > 0.9\#\mathbf{Cut}$ holds), then the component is classified specifically as the continuation of the segment.

If no component is specifically classified as the continuation, then check if one could be found with the following criteria: (1) the angle between the component and the segment is small (e.g., under 25°) and the base is most of the cut region ($\#\mathbf{B}_i > 0.9\#\mathbf{Cut}$ holds); (2) the component's base is the largest of the components whose angle between the segment is small (under, say, 25°); or (3) the component's angle between segment is the smallest and the angle is small (e.g., under 30°).

To estimate the angles between the segment and the components, define a segment region **S** consisting of the last $2N$ layers added into **Seg**. Then apply the techniques of Section 2.6 to **S** and $\mathbf{C}_i$ or use vectors connecting their top and bottom layers to estimate their directions.

If there are components classified as branch, proceed to step 5, otherwise proceed to step 6.

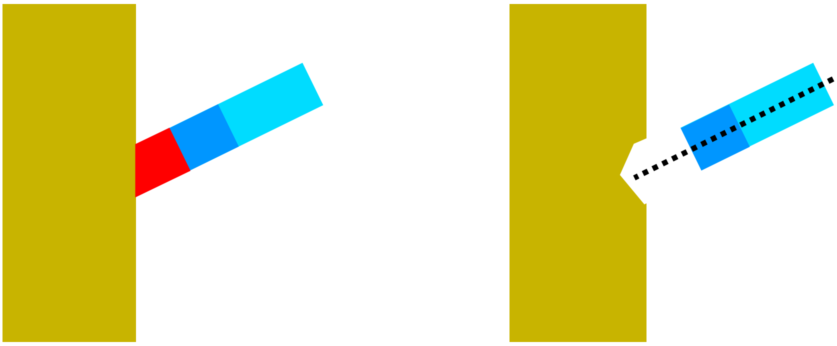
A5. Save New Segment Bases

For each component $\mathbf{C}_i$ classified as a branch, save its base $\mathbf{B}_i$ as the base of a new segment that is defined later with this same segmentation process. Do also the following updates:

$\mathbf{Cut} = \mathbf{Cut} \setminus \mathbf{B}_i$
$\mathbf{ForbSeg} = \mathbf{ForbSeg} \cup \mathbf{C}_i$

Next, **Seg** can be modified by the base $\mathbf{B}_i$. Especially in a branch that is thick compared to its parent, the cut region may be quite far in the branch before the study region becomes disconnected. Hence, there may be a significant appendage in the segment (see Figure A1). This appendage should be removed from the segment so that it does not affect the cylinder fitting later. This can be done by defining and removing the backward extended branch from the segment, *i.e.*, those cover sets whose distance to the estimated extended axis of the branch is about the same as the estimated radii of the branch (see Figure A1). Those parts taken out from the segment are added into $\mathbf{ForbSeg}$. If the removing of an appendage leaves a clear gap between fitted cylinders, it can be filled either by adding a cylinder (see Section 2.10) or expanding backwards the cylinder of the child branch.

Figure A1. Modfying the segment. Remove the appendage (red) resulting from the base of the branch (blue) from the segment. The black dotted line shows the axis of the branch.



When all the bases are saved and the segment is modified, proceed to step 6.

A6. Prepare for New Iteration

If **Cut** is not empty, it becomes a new previous cut region and the current segment is updated:

$\mathbf{Seg} = \mathbf{Seg} \cup \mathbf{Cut}$
$\mathbf{PreCut} = \mathbf{Cut}$

Then continue from step 2.
If **Cut** is empty, the current segment **Seg** is finished and use it to update the forbidden segments:

$\mathbf{Forb} = \mathbf{Forb} \cup \mathbf{Seg}$

Next, if new bases of branches not yet segmented exist for this tree component, choose the **Base**, which was created first, for the next round of segmentation and continue from step 2. with the following updates:

$$\mathbf{Seg} = \mathbf{Base}$$
$$\mathbf{PreCut} = \mathbf{Base}$$
$$\mathbf{Forb} = \mathbf{Forb} \cup \mathbf{Base}$$
$$\mathbf{ForbSeg} = \mathbf{Forb}$$

If no new bases of branches exist, the segmentation of the current tree component is finished. Then, if there are tree components not yet segmented, select the **TBase** of one of those components and continue from step 1. On the other hand, if all the tree components are segmented, the segmentation process is complete.

Article III

Analysis of geometric primitives in quantitative structure models of tree stems

Åkerblom, M., Raumonen, P., Kaasalainen, M.,
and Casella, E

Remote Sensing, 7(4):4581–4603.

DOI: http://dx.doi.org/10.3390/rs70404581

2015

Remote Sens. **2015**, *7*, 4581-4603; doi:10.3390/rs70404581

OPEN ACCESS

remote sensing

ISSN 2072-4292

www.mdpi.com/journal/remotesensing

Article

Analysis of Geometric Primitives in Quantitative Structure Models of Tree Stems

Markku Åkerblom [1],*, Pasi Raumonen [1], Mikko Kaasalainen [1] and Eric Casella [2]

[1] Department of Mathematics, Tampere University of Technology, P.O. Box 553, Tampere 33101, Finland; E-Mails: Pasi.Raumonen@tut.fi (P.R.); Mikko.Kaasalainen@tut.fi (M.K.)

[2] Centre for Sustainable Forestry and Climate Change, Forest Research, Farnham GU10 4LH, UK; E-Mail: eric.casella@forestry.gsi.gov.uk

* Author to whom correspondence should be addressed; E-Mail: markku.akerblom@tut.fi

Academic Editors: Nicolas Baghdadi and Prasad S. Thenkabail

Received: 3 February 2015 / Accepted: 3 April 2015 / Published: 16 April 2015

Abstract: One way to model a tree is to use a collection of geometric primitives to represent the surface and topology of the stem and branches of a tree. The circular cylinder is often used as the geometric primitive, but it is not the only possible choice. We investigate various geometric primitives and modelling schemes, discuss their properties and give practical estimates for expected modelling errors associated with the primitives. We find that the circular cylinder is the most robust primitive in the sense of a well-bounded volumetric modelling error, even with noise and gaps in the data. Its use does not cause errors significantly larger than those with more complex primitives, while the latter are much more sensitive to data quality. However, in some cases, a hybrid approach with more complex primitives for the stem is useful.

Keywords: tree modelling; terrestrial laser scanning; shape fitting; biomass estimation; error analysis

1. Introduction

The reconstruction of precise structure models of trees is important in understanding tree growth and decay, competition for resources, soil processes, photosynthesis, *etc*. The models can be used to compute, e.g., volumes, biomass estimates and size distributions. Terrestrial laser scanning (TLS) offers fast and accurate point clouds from which plant geometry can be reconstructed.

Many possible reconstruction methods from TLS data have been presented [1]. These methods can be divided roughly into three classes according to the way the tree structure is modelled. One class comprises the voxel methods, where volumetric models are constructed by partitioning the point cloud into voxels [2–4]. However, the ability of the voxel methods to model the tree structure is limited. The second class includes the parametric surface methods, where a single continuous surface is used to present a single branch [5,6]. In some of these methods, the cross-section of a stem or branch can be modelled as splines or other parameterizable curves [7]. The third class, and the one that we focus on, consists of shape-fitting methods, which use a geometric primitive to represent a part of a branch. A branch is modelled as a collection of these primitives [8,9]. Thies *et al.* [10] have presented a method that uses overlapping cylinder primitives to assess stem properties, and we have previously presented a reconstruction process based on geometric primitives [11].

Most of the published shape-fitting methods use the (rectangular) circular cylinder as their elementary building block. This choice is a compromise between simplicity and realistic reconstruction, as it is well known that the cross-section boundary of stems is never exactly a circle. For example, when the largest possible circle is fitted inside the cross-sections of eucalyptus trees, as much as 7% of the area is left outside the circle [12]. On average, the ratio between the minor and major diameters of various coniferous tree species is 0.96, indicating some elliptic features [13]. For branches, the ratio is expected to be even smaller due to gravity. Thus, it is a fair question if the use of circular cylinders is acceptable in such cases and if other shapes, that *a priori* appear more suitable, could be used instead with better results. Here, we study how the choice of the geometric primitive affects the model error and robustness of the reconstruction process and what are the pros and cons of using different shapes. In addition to the circular cylinder (**circyl**), we study the elliptical cylinder (**ellcyl**), the circular cone (**cone**) and the polygonal cylinder (**polcyl**) as the possible elementary blocks or geometric primitives (see Figure 1). In addition, we use polyhedral cylinder surfaces (**trian**), which are triangulated meshes, to reconstruct tree stems. Similar triangulated surfaces have been previously used for reconstruction with data collected with a digitizer from tree stems [14] and root systems [15] and for TLS data and root systems [16].

There is also another intuitive objection against the use of cylinders in particular, but also other geometric primitives: blocks or primitives cannot be attached to each other, such that the resulting surface is continuous and without gaps. However, we will show that, for most purposes, this is only an aesthetic problem, as all of the structural and geometric properties, such as the branching structure, branching angles, tapering, volumes, lengths, curvatures, *etc.*, can be modelled accurately. Indeed, if all we need are the structural and geometric properties, it is actually better to leave out continuity (and the related details) altogether in the modelling process, as these would only make it more difficult, slower and less reliable.

In the building-block approach, the first step is some segmentation of the point cloud into parts to which the geometric primitives are fitted. The segmentation process described in [11] was used in this study. This method divides the TLS point cloud efficiently into small surface patches that conform to the tree surface. These sets are used to segment the point cloud into branches and each segment further into sub-segments that are then reconstructed using shape fitting. In this study, mainly the tree stems are considered when testing the different reconstruction schemes, because of better data quality and the easy assessment of the results based on visualizations. However, in Section 4.5, we will briefly demonstrate how the approaches work with branches.

Shape fitting geometric primitives with their basic properties are presented in Section 2. The main reconstruction results presented in Sections 4–5 are discussed in relation to other works in Section 6 and presented in more compact form in Section 7. Some of the fitting problems considered here require strong initial values for the iterative fitting procedures to converge correctly, and the rest require meaningful, accurate parameters for usable results. In Section 3, procedures for finding good initial values and parameters from the data are presented, and the sensitivity of the fitting problems on the initial values is studied. In Section 4, we use generated stem models and simulated laser-scanning to study the error related to the shapes, as well as the effects of data quality. The simulated shapes are not meant to represent any particular trees; rather, they were chosen to portray various geometric characteristics in a somewhat exaggerated manner to analyse the performance and inherent properties of each geometric primitive. In Section 5, real field data are used in further tests. As no volume estimates are available for the field data, we use reconstructed models as references in Section 5.1. Simulated laser scanning is carried out for the reference models, and the resulting point clouds are reconstructed for the second time, in order to show how the approaches work with more realistic stem models.

2. Shapes

Here, we consider various geometric shapes suitable as geometric primitives. The shapes are shown in Figure 1, and they are described in Sections 2.1 to 2.5. Cylinders and cones have a few common parameters: the starting point $\mathbf{p}$ in three dimensions, the axis direction $\mathbf{a}$ (unit vector) and the length h in the axis direction. The shapes are rectangular, so the top and bottom planes are perpendicular to the axis $\mathbf{a}$.

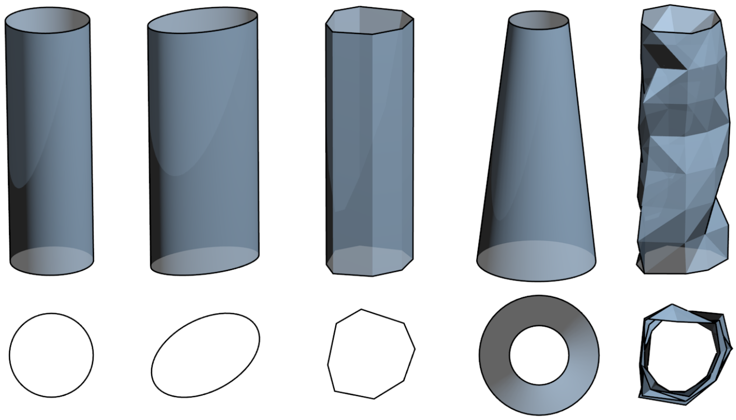


Figure 1. Geometric primitives. From left to right: circular cylinder (**circyl**), elliptic cylinder (**ellcyl**), polygon cylinder (**polcyl**), truncated circular cone (**cone**) and polyhedron (**trian**). (**Top**) Perspective side view; (**bottom**) orthographic top view.

2.1. Circular Cylinder

The simplest reconstruction shape is the circular cylinder (**circyl**), which requires only one additional parameter: the radius r. When using a **circyl**, the cross-section of a fraction of a branch is approximated as a disk with the radius r. For the **circyl**, the envelope area $E = 2\pi rh$ and volume $V = \pi r^2 h$. The parameters of each **circyl** are iteratively fitted to the dataset, as described in [11]. Each cylinder fitting is done iteratively starting from initial values. For cylinder and cone fitting details and parametrization, see [17].

2.2. Elliptic Cylinder

The elliptic cylinder (**ellcyl**) approximates the cross-section as an ellipse. Three additional parameters are used to model the ellipse: a pair of semiaxes radii (r_1, r_2) and the direction α of the major semiaxis (measured as the angle from a reference direction). We use the following approximation for the envelope area:

$$E \approx 2\pi h\sqrt{\frac{r_1^2 + r_2^2}{2}}. \tag{1}$$

The volume V can be computed accurately $V = \pi r_1 r_2 h$.

As a fitting problem when compared to the **circyl**, the additional parameters make the **ellcyl** more complex and time consuming. Therefore, we implement **ellcyl** fitting in two steps:

1. Estimate axis direction by fitting a **circyl**.
2. Project the points onto a plane perpendicular to the axis direction. Fit an ellipse to the two-dimensional data.

Since the steps are done in sequence, the procedure is not fully iterative.

2.3. Polygon Cylinder

The cross-section of a cylinder can be defined by any closed curve. However, it is sensible to keep the shape complexity to a minimum to achieve reasonable robustness and computational cost. Therefore, we use a polygon, which is a closed circuit of vertices $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k, \mathbf{v}_1 \in \mathbb{R}^3$ connected by straight lines, that in this case is assumed not to intersect itself. In a polygon cylinder (**polcyl**), the sequence $(\,\mathbf{v}_i\,)_{i=1}^{k}$ defines the base of the cylinder and the sequence $(\,\mathbf{v}_i + \mathbf{a}h\,)_{i=1}^{k}$ the top of the cylinder.

The perimeter p of a polygon is the sum of distances between consecutive vertices and the envelope area $E = ph$. The area of the polygon can be found by projecting the vertices onto a plane orthogonally and by using Gauss' area formula for these 2D points $(\tilde{\mathbf{v}}_i)$:

$$A = \left| \sum_{i=1}^{k-1} \frac{\tilde{v}_{i,1}\tilde{v}_{i+1,2} - \tilde{v}_{i,2}\tilde{v}_{i+1,1}}{2} \right|, \tag{2}$$

where $\tilde{v}_{i,1}$ and $\tilde{v}_{i,2}$ are the first and second component of the the vertex i, respectively. The **polcyl** volume $V = Ah$.

To fit a polygon cylinder to a point cloud $P = \{\,\mathbf{p}_i \in \mathbb{R}^3\,\}$, we propose the following algorithm.

1. Estimate the axis direction $\mathbf{a}$ either by the approaches described in Section 3 or by fitting a **circyl**.
2. Project the points onto a plane perpendicular to the axis $\mathbf{a}$. The projected set is denoted by $P_{\perp\mathbf{a}}$.
3. Compute a centre point $\mathbf{c}$ for the set $P_{\perp\mathbf{a}}$ either by computing the mean or by least-squares circle fitting [18].
4. Divide the elements of the set $P_{\perp\mathbf{a}}$ into k sectors around the centre point $\mathbf{c}$.
5. For each sector i, compute the mean point $\tilde{\mathbf{v}}_i$. If a sector has no elements, interpolate using the next and previous non-empty sectors.
6. Transform the points $(\,\tilde{\mathbf{v}}_i\,)$ back to the original coordinate system. The transformed sequence $(\,\mathbf{v}_i\,)$ is the base of the reconstructed **polcyl**.

2.4. Cones

A circular cone (**cone**) is a shape that is limited by a base circle and circular cross-sections that smoothly taper to a single point, the apex. A part of a branch is usually not the shape of a complete cone with an apex, but a truncated cone where the top is cut off. The fitting problem, however, remains essentially the same as in the cylindrical case. Fitting a **cone** to 3D data is an extension of this, since a cone has a minimum of six parameters [17]. There are two common ways to parametrize a circular truncated cone, either by using the taper angle or a pair of radii. Here, the latter is chosen: r_1 is the radius of the base, and r_2 is the radius of the top. In that case, the surface area $E = \pi(r_1+r_2)\sqrt{h^2+(r_2-r_1)^2}$ and the enclosed volume $V = \frac{1}{3}\pi h(r_1^2+r_1r_2+r_2^2)$. Cone fitting is done iteratively starting from initial values.

2.5. Polyhedron/Triangulation

A polyhedron is a considerably more general and flexible shape than cylinders or cones. In principle, the whole surface model of a tree could be given as a continuous polyhedron. However, due to insufficiencies in TLS data, it can be impossible to reliably reconstruct a tree as a single polyhedron surface, so sub-polyhedra as geometric primitives are an efficient means of interpolating between the gaps and for smoothing (regularizing) the effect of noise and outliers. Modelling a part of a tree as a polyhedron captures features in both the axial and radial directions. Polyhedron models are detailed and, compared to the other shapes, require more space for storing the data and are slower to compute.

Given an axis direction $\mathbf{a}$ and a reference point $\mathbf{p}$, a point cloud $P \subset \mathbb{R}^3$ can be partitioned into cells the following way:

1. In the axis direction, divide the data into layers $(R_i)_{i=1,\ldots,M}$ of equal height.
2. Further divide each layer into sectors $(C_j)_{j=1,\ldots,N}$.
3. A cell $P_{i,j} = R_i \cap C_j \subset P$.
4. Compute the centre point $\mathbf{v}_{i,j}$ of each cell (Figure 2, left).
5. Form triangle edges by connecting centre point pairs horizontally, vertically (Figure 2, centre) and diagonally (Figure 2, right).

If a cell is empty, its centre point is interpolated linearly using the vertices in the cells above and below. If interpolation is not possible in the axis direction, it can be done in the radial direction by using the previous and next non-empty vertices in the same layer of cells.

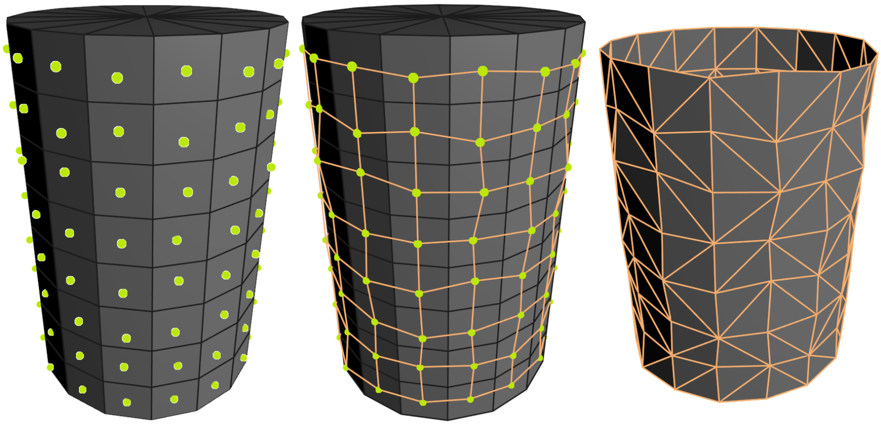


Figure 2. Stages of triangulation. (**Left**) Each vertex (green) is computed as the mean of points in each cell; (**centre**) neighbouring cells are connected by edges; (**right**) The quads are further triangulated, forming the polyhedron.

3. Initial Values

Next, we discuss initial estimates for shape parameters from point cloud data. These are needed in the iterative fitting problems, **circyl**, **ellcyl** and **cone**, as well as parameters for the partitioning in the **polcyl** and **trian** reconstructions.

Let S_i, $i = 1, \ldots, n$ be the subsegments of a segment, computed, e.g., as described in [11]. An axis point estimate $\mathbf{p}_i^*$ is computed as the mean of the subsegment points: $\mathbf{p}_i^* = \overline{\{\, \mathbf{x} \in S_i \,\}}$. There are different ways to estimate the axis direction $\mathbf{a}_i^*$, but we use the axis point estimates of consecutive subsegments: $\mathbf{a}_i^* = \mathbf{p}_{i+1}^* - \mathbf{p}_i^*$. This only works when there are at least two subsegments. For the last subsegment, we use the same direction as for second to last one. We have also presented alternative ways to estimate the axis direction; see Section 2.6 in [11]. Given the line defined by the axis point and direction estimates, the radius estimate r_i^* is the mean distance from the points of the subsegment to this line.

3.1. Sensitivity Analysis

To analyse the sensitivity of the shape fitting problems to their initial values, we used the curved cylinder model shown in Figure 3. TLS was simulated on the object, and reconstructions were computed from the point cloud with perturbations to the initial values. The appearance, volume and area of the resulting models were compared to the original.

For **circyl** and **cone**, all three initial values (radius, axis point, axis direction) are required, but **polcyl** and **trian** reconstructions do not need a radius estimate. **ellcyl** fitting was left out of the analysis, because the results are expected to be very similar to those of the **circyl**, due to the implementation. Furthermore, **cone** fitting also has the tapering parameter, but tests showed that when perturbed from 0 to 45°, the relative error in both the volume and area varied only in the magnitude of 10^{-3}. Note that the magnitude of the error is not as important as its variation under increasing perturbation.

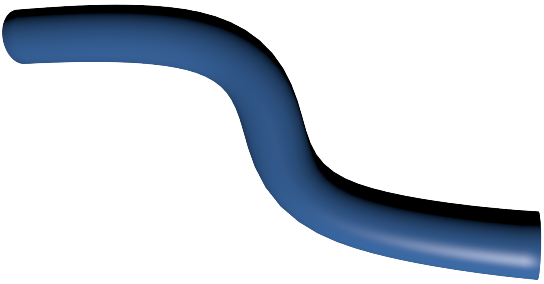


Figure 3. Curved pipe model used in the sensitivity analysis.

For the radius, perturbation was introduced by scaling the original estimate by factors from 0.1 to 2.5. The test showed that **circyl** and **cone** are both very indifferent to the radius estimate. With factors between 1.3 and 2.0, the perturbation had a minor effect on **cone** models, but the relative error stayed below 5% for both volume and area.

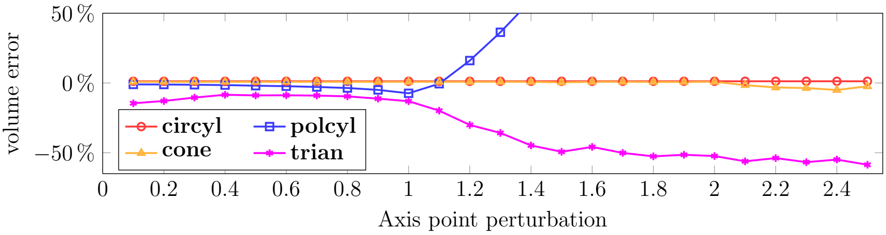


Figure 4. Relative volume error as a function of axis point estimate perturbation. The magnitude of the transition was received by multiplying the radius estimate by the scaling factor shown on the horizontal axis.

The axis point estimate was perturbed by moving it to a random direction perpendicular to the estimated axis direction. The magnitude of the move was the estimated radius scaled by a factor varied again from 0.1 to 2.5. The effect of the perturbation on the model volume is presented in Figure 4. For all of the shapes, the error remains small with scaling factor values lower than one. With larger values, as the axis point estimate moves outside the estimated cylinder hull, the error for **polcyl** starts to rise close to linearly for both quantities. At 2.5, the errors are 500% and 150%, for the volume and area, respectively. **trian** overestimates the area by ∼10% and underestimates the volume by up to 50% with factor values above 1.5.

The axis direction estimate was perturbed by rotating it in a random direction by an angle varying from 0° to 90°. The results for the volume are shown in Figure 5. The iterative fitting methods, **circyl** and **cone**, remain unaffected with perturbations smaller or equal to 45°. Even at 81°, the errors in the volume are only 5% and 7%, respectively. **polcyl** is very sensitive to the axis direction, even at small angle perturbations. With large perturbations, the **trian** model error can be up to 50% underestimation.

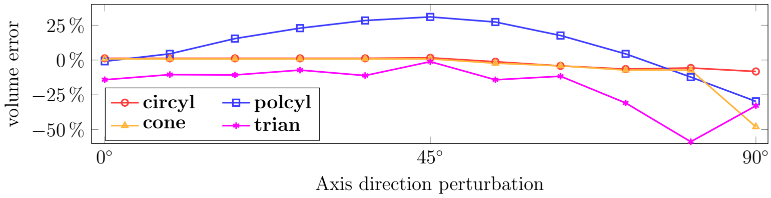


Figure 5. Relative volume error as a function of axis direction estimate perturbation.

Above, each of the initial values was studied individually. In real cases, the perturbation is never specific to a single property, but rather all of them.

4. Reconstruction from Generated Data

The reconstruction accuracy for the different shapes was studied using generated stem objects in order to better show the differences of the approaches. For this purpose, eight stem models were created using the 3D modelling software Blender. Some of the models are non-realistic, because the differences in the modelling approaches would not otherwise be visible. For a more realistic case, models derived from real oak trees were used in Section 5.1. The models were imported into MATLAB, where simulated TLS measurements were made. The resulting point clouds were reconstructed using all of the shapes.

4.1. Generation Process

The models (see Figure 6) are 15-m tall and consist of 302 consecutive 32-vertex rings. The diameter of the stems remains constant for the first 2 m at 0.40 m and then tapers smoothly to 0.12 m at the top. Stems 2–4 are unrealistically elliptic on purpose, and Stems 4, 6 and 8 have random perturbations on the surface.

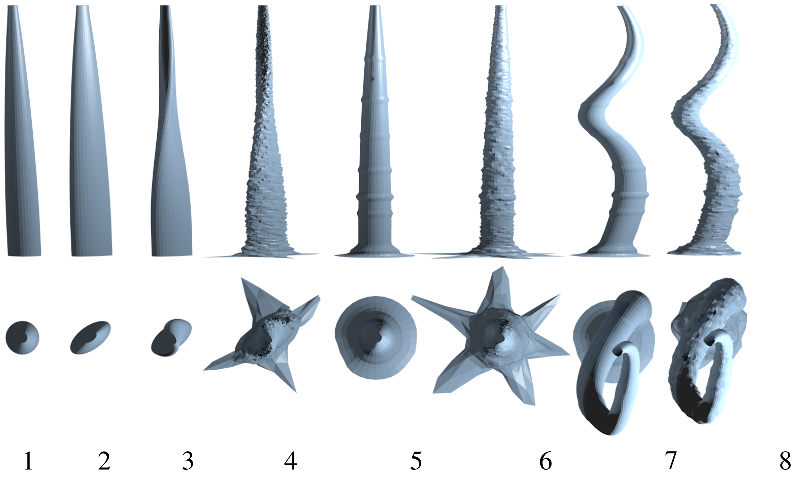


Figure 6. Flattened portraits of generated stem models (vertical dimension scaled down to 20%).

The simulated scans consisted of three positions around the model with an angular sampling resolution of 0.036° and a distance between the scanner and the models of ∼5 m. The scanner positions are visualized in Section 4.4. The scans were co-registered perfectly and were made with two noise levels: no noise and uniformly distributed noise between −3 mm and 3 mm.

4.2. Reconstruction Results

Because of randomness in the surface patch generation, the reconstruction was repeated 10 times for each shape to analyse the deviation σ_t associated with the underlying segmentation. The reconstruction accuracy of the different shapes was determined by comparing the total volume and surface area of the reconstructed model with the original one. Table 1 lists the average and maximum error for the volume and area of the noisy measurements and the volume of the noiseless measurements in relative error percentages. The standard deviations σ of different stem models and the average standard deviation σ_t associated with the segmentation randomness are also listed in percentage points. It is clear that the differences in the results between the noiseless and noisy measurements are very small, and thus, only the noisy data were used in further tests.

Table 1. Reconstruction results of generated stems. Maximum and average difference in percentages for the volume and surface area. Negative values stand for underestimation. The standard deviation of the average error (σ) and average of the standard deviations for the individual stems (σ_t) are also listed in percentage points (pp).

			circyl	**ellcyl**	**polcyl**	**cone**	**trian**
Noiseless	Volume	ΔV_{max}	−12.61	−19.20	−8.23	−24.59	−11.40
		ΔV_{avg}	1.22	0.34	−2.62	−5.13	−4.63
		σ	7.02	3.59	2.35	11.58	2.33
		σ_t	0.70	2.09	0.90	0.81	0.15
Noisy (±3 mm)	Volume	ΔV_{max}	−13.81	−19.20	−8.40	−24.43	−11.42
		ΔV_{avg}	1.36	0.20	−2.51	−5.00	−4.62
		σ	7.33	4.72	2.53	11.93	2.32
		σ_t	0.80	1.86	0.99	0.95	0.15
	Area	ΔA_{max}	−21.21	−17.83	−10.92	−26.04	−8.47
		ΔA_{avg}	−4.40	−0.96	−2.58	−7.20	−2.34
		σ	6.76	5.69	3.99	8.74	3.44
		σ_t	0.59	1.50	0.86	0.65	0.10

Stems 4 and 6 with the large flanges were the hardest to reconstruct accurately. Especially with the former, the combination of the flanges and highly elliptic cross-sections caused high errors in both volume and area. With the non-elliptic stems without flanges, reconstruction accuracy was much better for all of the shapes. The fitted cylinders and cones had an average length of 30 cm, whereas **trian** had a layer height of 10 cm.

With the **circyl**, all but the elliptic stems (2,3,4) were reconstructed with error between ±3.50% in the volume. In the surface area, the flanges in Stem 6 caused underestimation of over 7%, but with the other non-elliptic stems, the error was again between ±3%. The standard deviation σ seems relatively high, but it is heavily affected by the elliptic stems.

Using the **ellcyl** usually resulted in 2%–3% error in volume and area, either from over- or under-estimation. It did have the lowest average error, but also the highest average standard deviation σ_t. Overall, **ellcyl** seemed more unstable than the other approaches.

The **polcyl** approach seems to underestimate both the volume and the area by ∼2.5% on average. The largest error appears again in Stems 4 and 6, which have large flanges. The standard deviation of the average error is lower than with the iterative approaches, but slightly higher than with **trian**.

The large flanges were particularly hard for **cone** to model: too much tapering in the first cone caused errors of over 20%. On stems without flanges or elliptic cross-sections, the volume was underestimated only by ∼5% and the area by 4%. The standard deviation of the average error was the highest of all of the shapes.

The **trian** approach underestimates the volume of most stems by 3% to 5%, although Stem 6 causes higher error up to 10%. The surface area is reconstructed more accurately, with the average and maximum errors of −2.34% and 8.47%. The average standard deviation σ_t is considerably lower than with any other approach, and if Stem 6 is discarded, the maximum deviation is only 0.05 percentage points (pp). The standard deviation of the average errors σ was also the lowest of all. For taper curve comparison, see the Supplementary Material.

To illustrate how the error is distributed vertically within the models, Figure 7 shows the taper curve for Stem Model 7, and the respective error curves for each approach. Error curves are computed as the difference between the taper curves of the reconstructed model and the original one. The first meter is discarded in the figure for clearer visualization.

The taper curves obtained with different models are very similar along most of the stem. Larger differences only occur near the bulges of the original model. It seems that the **cone** models are affected the most by these fast variations in the original diameter, but none of the approaches can model the bulges correctly.

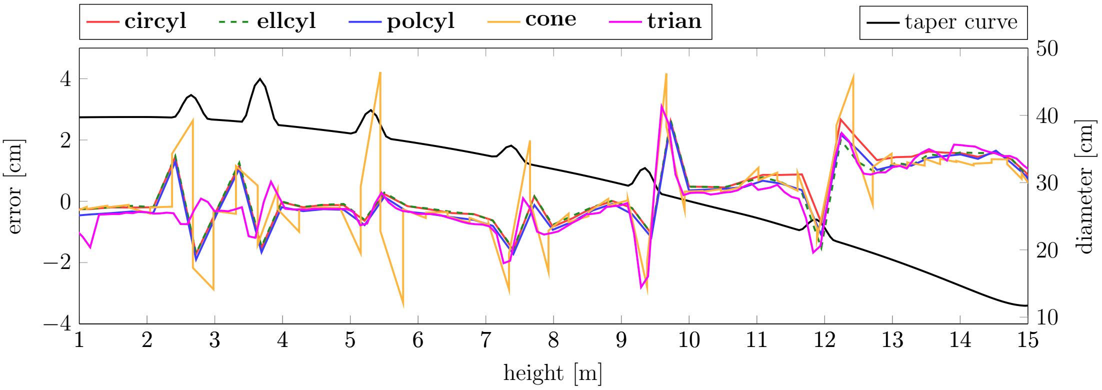


Figure 7. Taper error curves. Taper curve for the generated Stem 7 (right axis) and the diameter error (left axis) in the taper curves of the reconstructed models.

At the bottom half of the stem, where there are no bulges, the error level stays low and relatively stable. On the top half, *i.e.*, starting from 8 m, the trend of the error starts to rise almost linearly and turns from under- to over-estimation. This is due to the fact that as the diameter gets smaller, the noise becomes more relevant and also the sample density drops. At 15 m, the trend of the diameter error gets close to 2 cm.

Figure 8 shows how the different approaches are able to reconstruct the cross-sections along the vertical axis of Stem 8. The **circyl** seems to get the size of the cross-sections right, but is unable to capture the finer details. The **trian** approach captures the finer detail in the cross-sections, as it has better vertical resolution, but still noticeable differences remain.

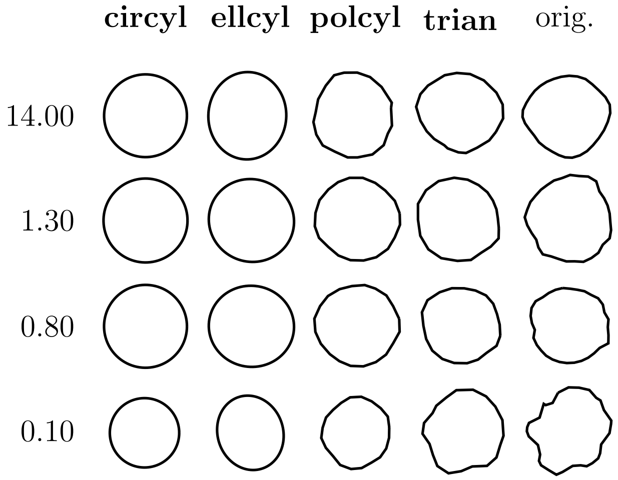


Figure 8. Cross-sections of the models of Stem 8 along the vertical axis, reconstructed from the noisy measurements.

4.3. *Effect of Shape Parameters*

The Stem Model 7 was used to test the effect of the length of the shapes on the reconstruction accuracy. The results showed that by using too short subsegments (less than 14 cm), the orientation could not be accurately determined, which resulted in overlapping primitives and, thus, overestimation in lengths and volumes. With larger lengths (up to 42 cm), the overestimation changed to slight underestimation, but was less than 5% for **circyl** and **ellcyl** and less than 8% for **polcyl**, and for **cone**, the volume error was close to 15%.

The effect of **polcyl** and **trian** vertex count was studied by reconstructing Stem 7 with the vertex count varying from four to 60. The results are visualized in Figure 9. Increasing the vertex count makes the reconstructions more accurate for both shapes. Already with 20 vertices, the reconstructed volume is underestimated by less than 10% and 3% with the **polcyl** and **trian**, respectively.

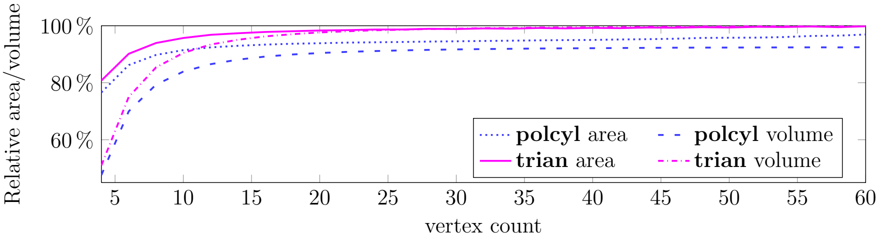


Figure 9. Effect of vertex count on **polcyl** and **trian**.

4.4. Data Quality

There are a number of factors that contribute to the quality of the scanning data and affect the reconstruction result. We tested how the different shapes handle changes in the sampling resolution, occluded parts and the number of scan positions. Because all of the approaches remain quite unaffected by a realistic amount of measurement noise, its effects were not investigated further. Similarly, registration errors and environmental factors, e.g., wind, were ignored.

The effect of the scanning resolution is shown in Figure 10 for the Stem Model 7. The resolution varied from 0.018° to 0.360°. Changing the point density forced us to change the parameters of the underlying segmentation process manually for each resolution, making the testing very tedious and probably not definitive. All of the cylinder-based approaches behaved quite similarly: slight underestimation with the finest resolutions and up to 10% overestimation with the crudest resolutions in the volume. With the area, the errors were even smaller, e.g., with 0.036° resolution, the **circyl** had an error of −6.54%. The underestimation in **cone** models increased steadily and was close to 18% at the end.

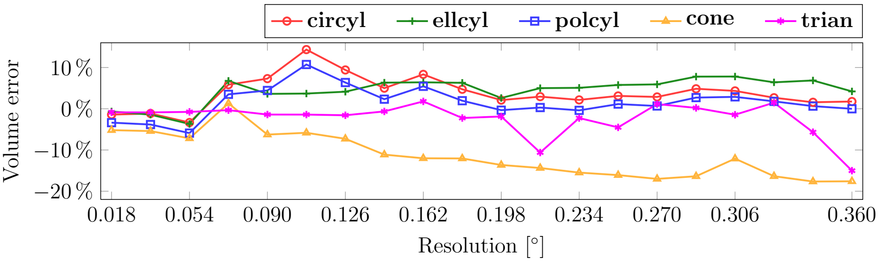


Figure 10. Effect of the angular sampling resolution on the reconstructed surface area and volume of Stem 7.

The effects of local occlusions were studied by iteratively degenerating the point cloud of Stem 7 scanned with a resolution of 0.036° by removing spherical sections. The radius of the sections varied linearly from 16 cm at the bottom to 8 cm at the top of the stem. The extractions we distributed vertically in a way that more of them were in the upper parts of the stem. The number of spherical extractions was increased from zero to 4850 with a total of 81 steps. Figure 11 shows examples of the four steps. Reconstruction was repeated 20 times for each step, and the resulting volume errors were averaged. The relative volume errors of the reconstructions as a function of the number of extractions is visualized in Figure 12.

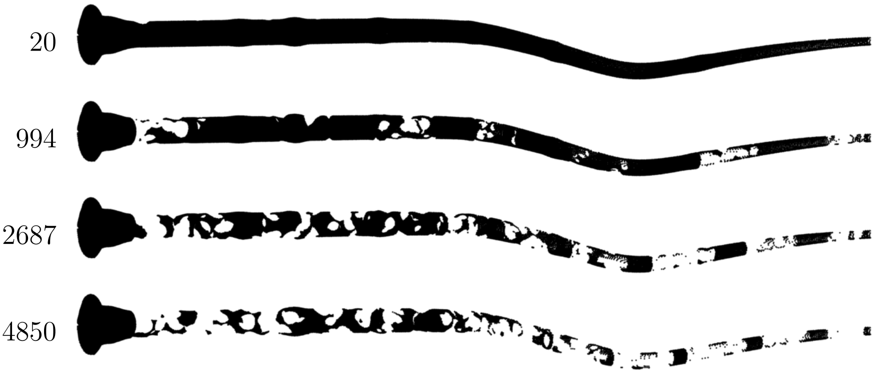


Figure 11. Example point clouds with increasing number of spherical section removals.

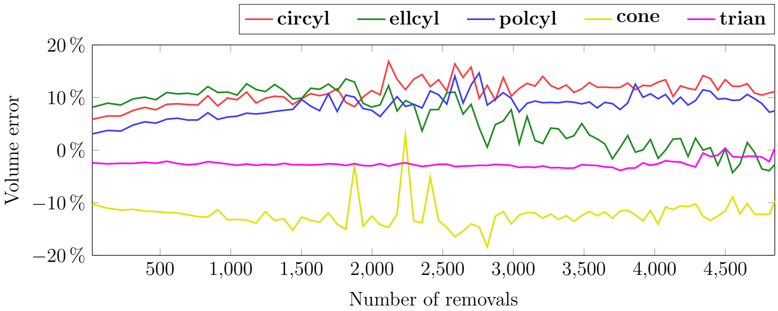


Figure 12. Effect of decreasing data quality on reconstructed volume. The horizontal axis represents the increasing number of spherical section removals from the point cloud scanned with the 0.036° sampling resolution.

Interestingly, **trian** was here the most stable approach, probably because it interpolated vertices into empty cells. The stability is weaker for the surface area: from about 3500 removals to 4850, the surface area error went from -1.3% to $+6.0\%$. For the other shapes, the results were similar for both the area and volume. **ellcyl** is the most unstable of the five approaches: after about 2000 removals, the volume starts to decrease due to ellipse fittings not converging and using initial values that are visibly too small. With the other shapes, **circyl**, **polcyl** and **cone**, the volume error increases slightly from zero to 2000 and remains relatively stable after that at $+10\%$, $+10\%$ and -10%, respectively.

In earlier tests, the object was always scanned all-around from three positions. This is not always possible, so the effect of the number of scanning positions was studied by using all possible combinations of the three directions visualized in Figure 13. The results are listed in Table 2 for Stem Model 7.

With a single scanning direction, all of the approaches produce high errors above 10%, at least for some directions. **trian** produces physically impossible results (negative volume) with a single scan.

With two scanning directions, the methods apart from **trian** produce quite accurate results. The **trian** approach still gives errors of several tens of percents. **circyl** has the smallest deviation and a mean closest to zero in these three cases. With all three directions, all of the approaches work well.

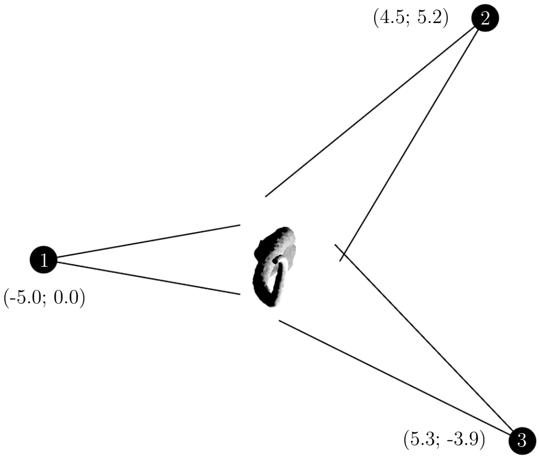


Figure 13. Simulated scanner setup. Stem Model 7 is visualized as the object. $(x; y)$-coordinates given for the positions. At each position, the scanner is at 1.5 m above the ground.

This stability test is of particular importance when considering fast scans of entire plots (from a few scanning positions) rather than detailed ones of individual trees. In such cases, the robustness of **circyl** (*i.e.*, its strong regularization) may be a crucial factor.

Table 2. Effect of the number of scanning positions for Stem Model 7. First column: included scanning directions. Others: relative volume error percentages.

Pos.	**circyl**	**ellcyl**	**polcyl**	**cone**	**trian**
1	2.63	−19.80	14.62	−7.38	−104.51
2	19.66	3.68	20.69	−2.02	31.18
3	3.28	−0.44	12.19	−12.46	−121.47
1,2	−2.26	0.20	−1.37	−8.18	31.66
1,3	−1.44	0.38	−5.29	−7.32	−26.72
2,3	−1.78	6.45	−0.73	−8.21	43.05
1,2,3	−2.62	−1.04	−5.58	−7.46	−1.43

4.5. Reconstruction of Complete Trees

To test how the approaches reconstruct branches, we generated a tree model with 40,600 triangles visualized in Figure 14. Laser scanning was simulated from the same positions as before in Figure 13 and the same resolution $0.036°$ and simulated measurement noise (uniform ±3 mm). The resulting point cloud consisted of 108,000 points. The tree was reconstructed 10 times with all of the approaches and compared to the known volume and area of the model. When the fitting of either the initial cylinder or the final shape did not converge, the cylinder defined by the initial values was used. Thus, parts of the **ellcyl**, **polcyl** and **cone** models are circular cylinders.

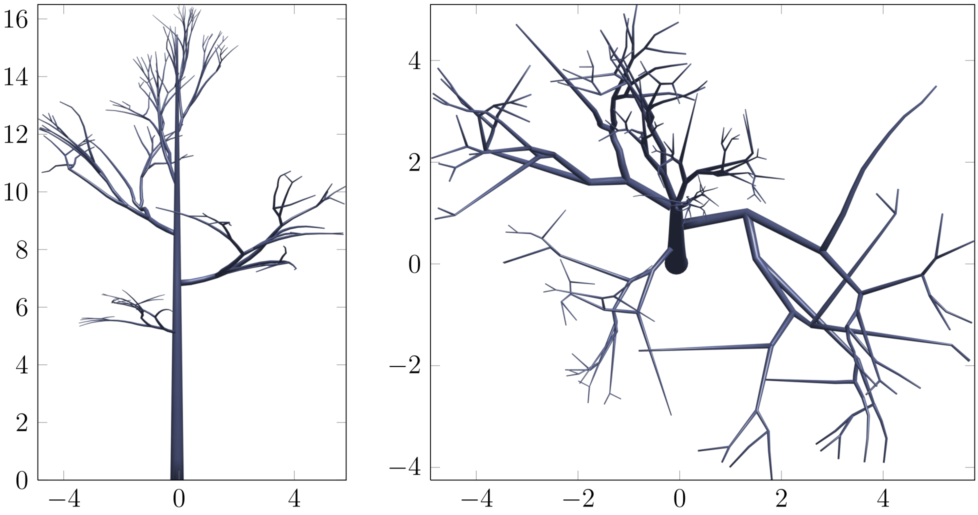


Figure 14. Generated tree model. (**Left**) Front view; (**right**) top view.

The average normalized volume and standard deviations are listed in Table 3. The results show that the iterative methods have average and maximum volume underestimation errors below 4%. Their standard deviation is also close to just one pp. **polcyl** overestimates the volume by a few percentages, where as the **trian** approach produces an overestimation error of over 50% on average. The high error level is caused by the lack of point coverage on the top parts of the branches. For example models and data tables, see the Supplementary Material.

Table 3. Reconstruction results for the generated tree model. Maximum and average difference in percentages for the volume and surface area. Negative values stand for underestimation. The standard deviation (σ_t) in percentage points (pp).

		circyl	**ellcyl**	**polcyl**	**cone**	**trian**
Volume	$\Delta V_{\max}$	−1.31	−2.71	7.47	−5.32	328.09
	ΔV_{avg}	−0.28	−0.75	2.93	−3.25	56.72
	σ_t	0.65	1.14	2.35	1.83	93.43
Area	$\Delta A_{\max}$	−4.89	−5.05	−2.15	−8.07	126.76
	ΔA_{avg}	−3.52	−2.29	0.13	−5.90	27.90
	σ_t	0.90	1.89	1.07	1.32	45.16

5. Reconstruction from Measured Data

To test the modelling approaches with measured data, scans were conducted in an 80-year-old oak plantation located in southeastern England (51.1533°N, 0.8512°W, 80 m asl). The plantation covers a total ground area of about 85 ha managed as a commercial forest of mixed species dominated at 75% by *Quercus robur* (English oak). A 30 × 30 m experimental area was selected and the understory removed before data acquisition. TLS data were recorded at nine positions (see Figure 15) over the experimental area in April 2012 (leaf-off). The TLS used was the single return HDS-6100 (Leica Geosystems Ltd., Heerbrugg, Switzerland).

In this study, we used scans from the bottom 14 m of eight oak stems, with a sampling resolution of 0.036°, and six of the eight scanning positions with the best visibility were selected for each stem. A thinned-out point cloud and reconstructed models of one of the stems, Stem 5, are shown in Figure 16.

The diameter at breast height (dbh) was manually measured using a girth tape, and it was compared to the reconstructed values. The average and maximum error percentages in the reconstructed dbh are listed in Table 4, together with the average standard deviation σ_t over 10 repeats within single stems, the deviation σ between the stems and the average distance from the measurements to the model surface. The dbh of the stems ranged from 27.7 cm to 41.6 cm.

The **cone** models have the highest average error close to 12% and the highest deviation. Because approximately the first 1.5 m is modelled as a single cone, the tapering of that cone is usually overestimated due to the bulging at the very bottom of the stems. The accuracy of the dbh measurements could be further improved for all of the cylinder and cone models, if a block were fitted to a short section over the breast height. The **trian** models have high errors, as well, but the underestimation is systematic. Because of the finer vertical resolution of **trian** models, the error in the diameter could be caused by the offset between the zero point of the model and the field ground level. The same systematic error could affect other shapes more, if they were shorter. The average distance between the measurements and the model surface was smallest with the **trian** and **polcyl** models, as expected, since they have the largest number of parameters.

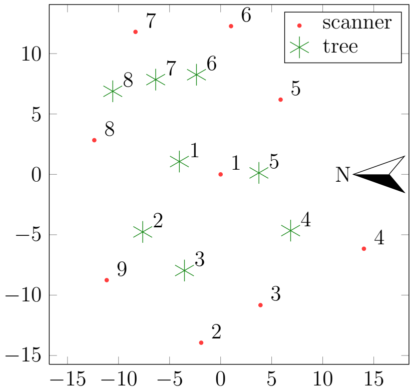


Figure 15. Scanner and tree locations. The scale is in meters.

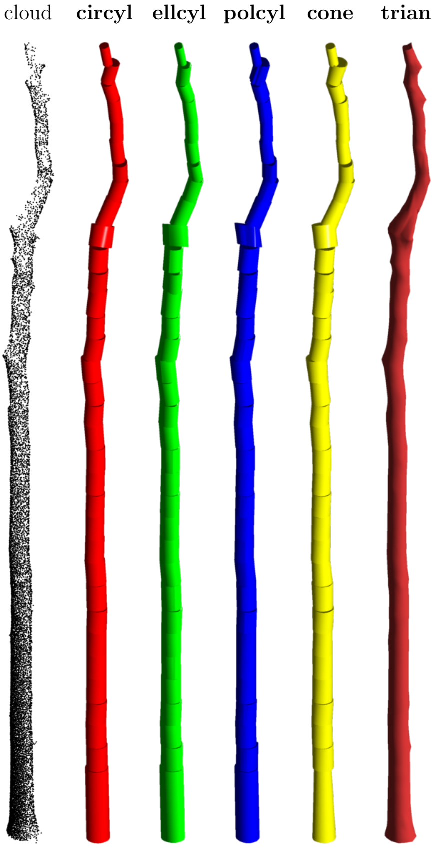


Figure 16. Point cloud (thinned) and reconstructed models of measured Stem 5.

Table 4. Reconstruction results for the measured stems. Maximum and average dbh error in percentages and standard deviation (σ), the average of the standard deviations for the individual stems (σ_t) in pp and the average distance d_avg from the measurements to the model surface in cm.

	circyl	ellcyl	polcyl	cone	trian
ΔD_max	−7.08	7.63	−7.38	−23.28	−6.69
ΔD_avg	−1.12	−0.41	−1.51	−11.57	−4.84
σ	1.94	1.94	1.83	6.19	1.39
σ_t	1.10	1.56	1.04	2.30	0.01
d_avg	1.19	1.07	0.88	1.11	0.71

In the bottom half of the stems, the ratio of the minor and major radii of the fitted elliptic cylinders for all the stems was 0.92, and the variance was small. On the upper half, the ratio started to decline, and the variance was four-times larger. The same effect was visible for the tapering angle of the fitted cones. The visualizations of the models showed that unrealistically tapering cones and elliptic cylinders were fitted to the upper parts of the stems. On average, the ellipse radii ratio was 0.87 and the cone tapering angle $0.33°$. Thus, additional constrains should be introduced when using these shapes for stem parts with high noise or a low number of returns.

5.1. Second Order Reconstructions

To better estimate the volume and area error levels and distribution, the following procedure was used. The original point clouds were used to reconstruct reference **trian** models, with 32 vertices, for the stems respectively. Simulated TLS was then carried out with the same settings and scanner locations as in Section 4, except that no additional noise was added to the measurements. The resulting point clouds were then used to compute second order reconstructions ten times with all of the shapes, including **trian** with 20 vertices. The results are listed in Table 5, and they are in agreement with the results using generated stem models in Section 4. Here, the differences between the shapes are minimal, because the oak stems are simpler than the generated stem models. The average volume underestimation for all of the shapes other than **cone** is below 3%. For taper curve comparison, see the Supplementary Material.

Table 5. Average and maximum volume and area errors (%) and standard deviations (pp) when second order reconstructions were compared to respective reference models.

		circyl	**ellcyl**	**polcyl**	**cone**	**trian**
Volume	ΔV_{max}	−6.69	−6.32	−8.96	−9.60	−7.76
	ΔV_{avg}	−1.33	−0.18	−2.30	−5.45	−2.72
	σ	2.87	3.68	2.99	2.17	2.28
	σ_{t}	0.37	2.62	0.98	1.08	0.49
Area	ΔA_{max}	−10.73	−9.32	−8.19	−12.67	−5.24
	ΔA_{avg}	−5.80	−3.97	−3.86	−8.04	−3.59
	σ	2.67	2.98	1.96	2.74	1.02
	σ_{t}	0.26	2.69	0.67	0.67	0.34

6. Discussion

The results show that in terms of robustness to data quality and initial values, the use of the circular cylinder as a geometric primitive in reconstruction approaches (e.g., [8–11]) is justified. The expected error in volume and surface are low even when comprehensive data are not available. However, the circular cylinder is not able to model the fine structure of the stem cross-section. For this purpose, the **trian** models are better, but they require comprehensive input data. The resulting models require a lot more disk space to be stored and slower computations, when comparing to the circular cylinder, which is sufficiently compact for massive-scale automatic reconstructions [19].

Simple eccentricity characteristics, e.g., the ratio between the largest and smallest diameter, can be computed from the **ellcyl** models, and for the English oak stems, the ratio was 0.92 (at lowest 3 m), which seems realistic when comparing to the value of 0.96 measured for various coniferous species [13]. Similar estimates could be computed from the **polcyl** and **trian** models. None of the presented structure models include information about the inner structure, e.g., pith eccentricity [20], and for that purpose, destructive measurements are still needed.

The problem with using more complex shapes is the increased requirement for measurement coverage around the object. Lack of coverage combined with shape fitting instability can lead to very high errors. Thus, it would be beneficial to analyse, e.g., the radial coverage locally around a given axis to determine the maximum complexity of the shape that should be fitted to those measurements. By implementing such local coverage analysis, the resulting models would consist of several types of geometric primitives. Especially, the lower part of stems has, in most cases, sufficient coverage for **polcyl** or **trian** reconstruction. Thus, the resulting model would consist of the stem polyhedron and a collection of cylinders for the top part of the stem and the branches. Another possibility is to make several reconstructions with different shapes for a single part and store all of them in the same model. Which representation to use could be determined later depending on the application (*cf.* [21]).

The data in Section 4 were generated using a combination of 3D modelling and simulated laser-scanning. By further developing and optimizing the scanning part, it would be possible to model entire forest plots with understory and to simulate scans containing multiple trees or even to simulate continuous scans with moving scanners. Furthermore, on the tree level, this type of simulation offers an inexpensive way to study how to model tree abnormalities, such as cankers, seams or buttress roots. This way, it is possible to start with ideal data without any noise or occlusion present and to degenerate them in various ways to get data similar to real measurements.

7. Conclusions

In this study, reconstruction approaches based on four different geometric primitives and a triangulation approach were tested with both measured and simulated data. For all of the shape fitting approaches, the sub-segmentation procedure presented [11] was used, but most of the results are applicable to any reconstruction procedures using the same geometric primitives [9,10]. Using simulated data provided correct reference values of quantities, such as the volume, surface area and taper curves. These also served as a way to determine how much of the error was caused by the modelling and how much by the quality of the data. The most noticeable differences in the modelling approaches occurred in their ability to adapt to different shapes of stems and in their robustness to the quality of data. Adding a realistic amount of measurement noise did not noticeably change the reconstruction results in terms of volume or surface area. None of the tested approaches could accurately model complex cross-sections, such as flanges. On the other hand, the polyhedron method has been shown to give a good representation of complicated stump shapes that are not well represented by the other primitives [16,22].

Circular cylinder The **circyl** primitive is the most commonly used [9–11]. It is robust, as it is very indifferent to perturbations in all of the initial values of the fitting. Even coarse data-based estimates converge towards the correct solution; e.g, the axis direction estimate can be up to 45° off without causing any error, and after that, the error was below 10%. For the generated stems, the average volume overestimation was 1.36% with an average standard deviation of 7.33 pps. The surface area was underestimated with an average of 4.40% ± 6.76 pp. With field data, dbh was underestimated by 1.12%, on average, with a standard deviation of 1.94 pp.

Elliptic cylinder The **ellcyl** fitting was implemented as a combination of **circyl** fitting and two-dimensional ellipse fitting to fine-tune the cross-section. Thus, in the initial values, it performed as **circyl**. With the generated data, it had the lowest average error at 0.20% for the volume and −0.96% for the area. The deviation between repeats was the highest at 1.86 pp for the volume and 1.50 pp for the area. The high deviation means that **ellcyl** fitting can be unpredictable, but implementing the fitting as a single iterative process might help stabilize it. If the data are comprehensive and have good quality, the **ellcyl** fitting can produce good results. Furthermore, **ellcyl**s fitted to the measured stems were elliptical rather than circular, as the ratio of the major and minor radii was 0.87 on average.

Polygon-based cylinder The **polcyl** reconstruction is not an iterative process, so accurate initial values are crucial. Even small perturbations to the axis direction estimate caused volume and area errors over 10%. However, in the tests with the generated and measured stems, the initial values received from the data were sufficient to produce error levels similar to the **circyl**, and when visualized, the models appeared accurate. On the generated stems, the average error in the volume was −2.51% ± 2.53 pp and on the area −2.58% ± 3.99 pp. As expected, the **polcyl** was more robust on the stem cross-section shape and reconstructed the elliptic stems more accurately than **circyl** and **cone**. With 26 vertices on the base polygon, the cylinder was able to reconstruct the dbh of the measured stems with an average diameter error of 7.38%.

Circular cone The **cone** handles initial value perturbations almost identically to the **circyl**. Using **cone** resulted in an average 5.00% ± 11.93 pp underestimation of the volume and 7.2% ± 8.74 pp of the area with the generated stems. It almost always resulted in the smallest volume of all of the models. Especially problematic was the bulging at the bottom of the stems, resulting in overestimation on the cone tapering. This caused high errors in the dbh values of the models. Most of the **cone** fitted to the measured stems tapered only a little, 0.33° on average. Thus, the difference in volume in comparison to **circyl** is not probably worth the increased instability, especially as the difference is expected to decrease with the primitive length. If one were to use longer cones, the stability and accuracy should be increased.

Polyhedron The **trian** approach is fairly sensitive to the values of the axis point and axis direction, of which the former had the larger effect on accuracy. It requires full coverage around the object: where the other primitives performed well with just two scans, it required all three positions or the volume error was close to 30%. On the other hand, it was not sensitive to local occlusions in the data. With the generated stem models, it produced a volume error of −4.62% ± 2.32 pp, and the error in the surface area was even lower. The average diameter error in dbh was −4.84%, but seemed systematic and could be caused by either the selection of breast height in the model or too small a number of vertices used.

Overall The results show that any of the models, even with cruder than normal resolution, can achieve accurate volume and area estimation. The average volume error for all models was below 5%, and using, e.g., the circular cylinder model on extremely elliptic stems, resulted in a maximum volume error below 14%. These ties well with the field-calibrated studies of [16,23], as the volume and length errors are similar in all studies. In [16], it is shown that a hybrid version of polyhedral and cylindrical modelling is accurate, even for the highly complex shapes of stump-root systems. Furthermore, [24] shows that using allometric equations to predict the volume of eucalyptus trees results in a 36.6% average error with the species-specific equation and a 29.9% error with the generic eucalyptus equation, while circular cylinder structure models produce an error of 9.7%.

Extrapolating the stem-based results to branches predicts that increased occlusions and relative noise levels and lower point density will increase the demand of robustness and the stability of the fitting scheme used. Thus, the general recipe is to use **circyl** for branches and, when necessary, more complex primitives for parts of the stem. However, the advantage gained by the latter is not significant, at least in terms of volume. For a fast and compact processing of a large number of trees, the **circyl** approach is rather sufficient. Taking into account the typical systematic errors and glitches that occur in field campaigns, a rule-of-thumb volumetric error from any surface modelling approach can be expected to be ±10%. The effect of the wind and co-registration error are expected to increase the error level, especially for the branches, but they were not considered in this study.

Acknowledgements

This study was funded by the Academy of Finland research projects Inverse Problems of Regular and Stochastic Surfaces and Centre of Excellence in Inverse Problems.

Author Contributions

Markku Åkerblom created the stem models for the generated data section, reconstructed the models for both data types, computed the results, created the figures and wrote the main text. Pasi Raumonen created the reconstruction procedure, guided the ideas of this manuscript, wrote parts of the text and gave feedback on it. Mikko Kaasalainen supervised the whole process from the idea-stage to the final manuscript, wrote parts of the text and gave vital feedback on rest of the text. Eric Casella carried-out the *in situ* measurements and provided the data used in Section 5, wrote the description of the measurements and gave feedback on the manuscript.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Dassot, M.; Constant, T.; Fournier, M. The use of terrestrial LiDAR technology in forest science: Application fields, benefits and challenges. *Ann. For. Sci.* **2011**, *68*, 959–974.
2. Gorte, B.; Pfeifer, N. Structuring laser-scanned trees using 3D mathematical morphology. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2004**, *35*, 929–933.
3. Lefsky, M.; McHale, M.R. Volume estimates of trees with complex architecture from terrestrial laser scanning. *J. Appl. Remote Sens.* **2008**, *2*, 023521–023521.
4. Vonderach, C.; Voegtle, T.; Adler, P. Voxel-base approach for estimating urban tree volume from terrestrial laser scanning data. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2012**, *39*, B8.
5. Xu, H.; Gossett, N.; Chen, B. Knowledge and heuristic-based modelling of laser-scanned trees. *ACM Trans. Graph.* **2007**, *26*, 19:1–19:13.
6. Yan, D.M.; Wintz, J.; Mourrain, B.; Wang, W.; Boudon, F.; Godin, C. Efficient and robust tree model reconstruction from laser scanned data points. In Proceedings of the 11th IEEE International conference on Computer-Aided Design and Computer Graphics, Huangshan, China, 19–21 August 2009; pp. 572–576.

7. Pfeifer, N.; Winterhalder, D. Modelling of tree cross sections from terrestrial laser scanning data with free-form curves. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2004**, *36*, W2.
8. Pfeifer, N.; Gorte, B.; Winterhalder, D. Automatic reconstruction of single trees from terrestrial laser scanner data. In Proceedings of the 20th ISPRS Congress, Istanbul, Turkey, 12–23 July 2004; pp. 114–119.
9. Cheng, Z.L.; Zhang, X.P.; Chen, B.Q. Simple reconstruction of tree branches from a single range image. *J. Comput. Sci. Technol.* **2007**, *22*, 846–858.
10. Thies, M.; Pfeifer, N.; Winterhalder, D.; Gorte, B. Three-dimensional reconstruction of stems for assessment of taper, sweep and lean based on laser scanning of standing trees. *Scand. J. For. Res.* **2004**, *19*, 571–581.
11. Raumonen, P.; Kaasalainen, M.; Åkerblom, M.; Kaasalainen, S.; Kaartinen, H.; Vastaranta, M.; Holopainen, M.; Disney, M.; Lewis, P. Fast automatic precision tree models from terrestrial laser scanner data. *Remote Sens.* **2013**, *5*, 491–520.
12. Medhurst, J.; Ottenschlaeger, M.; Wood, M.; Harwood, C.; Beadle, C.; Valencia, J.C. Stem eccentricity, crown dry mass distribution, and longitudinal growth strain of plantation-grown *Eucalyptus nitens* after thinning. *Can. J. For. Res.* **2011**, *41*, 2209–2218.
13. Tong, Q.; Zhang, S. Stem form variations in the natural stands of major commercial softwoods in eastern Canada. *For. Ecol. Manag.* **2008**, *256*, 1303 – 1310.
14. Yoshimoto, A.; Surový, P.; Konoshima, M.; Kurth, W. Constructing tree stem form from digitized surface measurements by a programming approach within discrete mathematics. *Trees* **2014**, *28*, 1577–1588.
15. Danjon, F.; Reubens, B. Assessing and analyzing 3D architecture of woody root systems, a review of methods and applications in tree and soil stability, resource acquisition and allocation. *Plant Soil* **2008**, *303*, 1–34.
16. Smith, A.L.; Astrup, R.; Raumonen, P.; Liski, J.; Krooks, A.; Kaasalainen, S.; Åkerblom, M.; Kaasalainen, M. Tree root system characterization and volume estimation by terrestrial laser scanning and quantitative structure modelling. *Forests* **2014**, *5*, 3274–3294.
17. Lukács, G.; Martin, R.; Marshall, D. Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. In *Computer Vision-ECCV'98*; Springer Berlin Heidelberg: Berlin, Germany, 1998; pp. 671–686.
18. Wolberg, J. *Data Analysis Using the Method of Least Squares: Extracting the Most Information from Experiments*; Springer-Verlag Berlin Heidelberg: Berlin, Germany, 2006.
19. Raumonen, P.; Casella, E.; Calders, K.; Murphy, S.; Åkerblom, M.; Kaasalainen, M. Massive-scale tree modelling from TLS data. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2015**, *II-3/W4*, 189–196.
20. Robertson, A. Centroid of wood density, bole eccentricity, and tree-ring width in relation to vector winds in wave forests. *Can. J. For. Res.* **1991**, *21*, 73–82.
21. Godin, C.; Caraglio, Y. A multiscale model of plant topological structures. *J. Theor. Boil.* **1998**, *191*, 1–46.

22. Liski, J.; Raumonen, P.; Kaasalainen, S.; Repo, A.; Krooks, A.; Akujärvi, A.; Kaasalainen, M. Indirect emissions of forest bioenergy: detailed modeling of stump-root systems. *Glob. Change Biol. Bioenergy* **2014**, *6*, 777–784.
23. Kaasalainen, S.; Krooks, A.; Liski, J.; Raumonen, P.; Kaartinen, H.; Kaasalainen, M.; Puttonen, E.; Anttila, K.; Mäkipää, R. Change detection of tree biomass with terrestrial laser scanning and quantitative structure modelling. *Remote Sens.* **2014**, *6*, 3906–3922.
24. Calders, K.; Newnham, G.; Burt, A.; Murphy, S.; Raumonen, P.; Herold, M.; Culvenor, D.; Avitabile, V.; Disney, M.; Armston, J.; *et al.* Non-destructive estimates of above-ground biomass using terrestrial laser scanning. *Methods Ecol. Evol.* **2014**, *6*, 198–208.

Article IV

Automatic tree species recognition with quantitative structure models

Åkerblom, M., Raumonen, P., Mäkipää, R., and Kaasalainen, M

Remote Sensing of Environment, 191:1 – 12.

DOI: http://dx.doi.org/10.1016/j.rse.2016.12.002

2017

Remote Sensing of Environment 191 (2017) 1–12

Contents lists available at ScienceDirect

Remote Sensing of Environment

journal homepage: www.elsevier.com/locate/rse

Automatic tree species recognition with quantitative structure models

Markku Åkerblom[a,*], Pasi Raumonen[a], Raisa Mäkipää[b], Mikko Kaasalainen[a]

[a]*Department of Mathematics, Tampere University of Technology, P.O. Box 553, 33101Tampere, Finland*
[b]*Natural Resources Institute Finland (Luke), P.O. Box 18, Vantaa FI-01301, Finland*

ARTICLE INFO

Article history:
Received 17 May 2016
Received in revised form 10 November 2016
Accepted 3 December 2016
Available online xxxx

Keywords:
Tree species recognition
Terrestrial laser scanning
Quantitative structure model
Tree reconstruction

ABSTRACT

We present three robust methods to accurately and automatically recognize tree species from terrestrial laser scanner data. The recognition is based on the use of quantitative structure tree models, which are hierarchical geometric primitive models accurately approximating the branching structure, geometry, and volume of the trees. Fifteen robust tree features are presented and tested with all different combinations for tree species classification. The classification methods presented are *k*-nearest neighbours, multinomial regression, and support vector machine based approaches. Three mainly single-species forest plots of Silver birch, Scots pine and Norway spruce, and two mixed-species forest plots located in Finland and a total number of trees over 1200 were used for demonstration. The results show that by using single-species forest plots for training and testing, it is possible to find a feature combination between 5 and 15 features, that results in an average classification accuracy above 93% for all the methods. For the preliminary mixed-species forest plot testing, accuracy was lower but the classification approach presented potential to generalize to more diverse cases. Moreover, the results show that the post-processing of terrestrial laser scanning data of multi-hectare forest, from tree extraction and modelling to species classification, can be done automatically.

1. Introduction

Large multi-hectare areas of forests with thousands of trees can now be measured quickly with terrestrial laser scanning (TLS) (Calders et al., 2015a). This kind of massive-scale remote sensing of trees requires that most, if not all, post-processing steps are done automatically. In addition to the geometric and volumetric data, an important piece of information that can be determined from the point clouds is the tree species. For example, the species have an effect on the greenhouse gas exchange of a tree (Meier et al., 2016), and measuring the change in biodiversity is related to the number of species and their distribution. Thus, automatic and reliable tree species recognition would be an essential step to make the massive-scale remote sensing from TLS data practical.

There are a number of published studies that use TLS data for tree species recognition: Haala et al. (2004) used the combination of TLS and high-resolution panoramic images to make a comparison of the bark textures of four trees. Their results show that the texture is a candidate for classification as it seems to stay similar in a stem, but differs between stems. However, the approach was not tested on a larger dataset nor was it automatic.

Puttonen et al. (2010) used TLS and hyperspectral data to classify 24 trees of three species with a support vector machine (SVM). The scanning was done indoors, so point cloud segmentation into trees was not required. The classification features included shape parameters computed from the TLS data and averaged reflectance values of the hyperspectral data. With a combination of 2 features from each dataset, the average classification accuracy was over 85% for all species. When using only a pair of TLS data features, the accuracy was over 70% for only 43% of the pairs, but the best classification accuracy was 95.8%.

Puttonen et al. (2011) combined mobile laser scanning (MLS) and hyperspectral data to classify 133 trees of 10 species with SVM. Individual trees were manually isolated from the point cloud. Similarly to Puttonen et al. (2010) , the classification features consisted of MLS-based shape parameters and per channel averaged spectral data. The results showed that MLS features on their own were able to separate coniferous and deciduous trees with 90.5%, and individual species with 65.4% accuracy. For the combination of MLS and spectra the percentages were 95.8% and 83.5%, respectively.

Vauhkonen et al. (2013) tested hyperspectral LiDAR (HSL) in laboratory conditions for classifying 18 spruce and pine trees. The classification accuracies varied between 78% and 89%. Different scans of the same trees were used for training and classification.

[*] Corresponding author.

http://dx.doi.org/10.1016/j.rse.2016.12.002
0034-4257/© 2016 Elsevier Inc. All rights reserved.

Othmani et al. (2013) used the 3D texture of the bark of 230 trees of five species. In their approach a 30 cm long patch is manually isolated in a stem and its texture analysed using 2D signal processing techniques and a random forest (RF) classifier. The overall species recognition was 88%.

In a most recent study (Lin and Herold, 2016), 40 trees of 4 species were classified by using SVM and *explicit tree structure* parameters (ETS). In contrast to the shape parameters used by, e.g., Puttonen et al. (2011) , ETS parameters describe the actual shape of the tree stem or crown rather than the distribution of TLS samples. The authors refer to tree isolation details in Holopainen et al. (2013) but fail to state which of the methods was used, and thus the level of automation is unknown. At least the separation of stem and branch points is done interactively. The classification tests were done using the leave-one-out cross-validation (LOOCV) in two different scenarios, maximum and robust, with accuracies 90.0% and 77.5%, respectively. The authors state that the latter scenario is more likely to be suitable for real applications.

The above literature survey shows that the tree species recognition from TLS data has been the topic of only a few studies and in most cases it has been combined with other data sources to achieve sufficient classification accuracies. Furthermore, the sample sizes have been relatively small, and no fully automatic solution has been presented yet.

In this paper, we propose a proof-of-concept for fully automatic species recognition approach from TLS measurements. Rather than using 3D point cloud data directly for classification, trees are first reconstructed as quantitative structure models (QSM) (Calders et al., 2015b; Raumonen et al., 2013). Notice that the QSM reconstruction is done by using only the *xyz*-coordinates of the points and thus no intensity data, spectral information, photographs, or ultra-high resolution scans are required. The classification features are computed from the geometric and topological tree properties stored in the models, which means that we have more than three dimensions to work with. This enables the use of properties that have been hard or impossible to determine directly from the point cloud data. The proposed classification features are listed in Section 2.5.

For the species recognition, we tested three different classification methods with numerous feature sets to show their differences and suitability for the application. Namely, we tested *k*-nearest neighbours, multinomial regression, and support vector machine based approaches. The classification methods are presented in Section 2.4.

It has been shown that QSMs can be automatically computed in massive scale (Raumonen et al., 2015), and when combined with automated feature computations it makes the complete classification procedure fully automatic. To demonstrate the approach, three large, mainly single-species plots from Finland are used. In addition, two mixed-species forest plots, also from Finland, are used to demonstrate preliminary results from more heterogeneous stands. The three species are the most numerous in Finland and represent both deciduous Silver birch (*Betula pendula* Roth) and coniferous Scots pine (*Pinus sylvestris* L.) and Norway spruce (*Picea abies* [L.] Karsten). Forest plot and scanning setup details are presented in Sections 2.1 and 2.2, respectively. We present results in Section 3, and sum up in Section 5.

2. Materials and methods

2.1. The forest plots used for the demonstration

We have three large almost single-species forest plots and two unmanaged mixed-species stands, which have been scanned with TLS. One of the single-species plots is a systemically planted plot with only Silver birch trees and the other two are natural coniferous plots with Norway spruce and Scots pine trees. All three study plots are in Punkaharju, Finland, where annual mean precipitation is 600 mm and effective temperature sum 1300 dd (Merilä et al., 2014).

2.1.1. Silver birch plot

A Silver birch stand used in this study is a field experiment in Punkaharju, Finland (61°48′N, 29°18′E), established in 1999 to study within-stand differences among genotypes (22 genotypes micropropagated from local trees) (Possen et al., 2014). Trees were planted on agricultural field with a planting distance of 2 × 2 m in 1999. In April 2008, 50% of the trees were harvested. At the time of the scanning in October 2014, the stand density was approximately 1000 trees per ha, height of the trees varied from 18 to 24 m, and diameter at the height of 1.3 m (DBH) was 10–17 cm.

2.1.2. Scots pine plot

The Scots pine dominated study plot in Punkaharju, Finland (61°46′N, 29°20′E) is conventionally managed forest. The latest thinning took place in 1994, thereafter only dead trees are removed. At the time of the TLS in October 2014, the stand age was 95 years, stem number was approximately 500 stems per ha, the DBH was 18–40 cm, and the height of trees 27–32 m. The stand grows on sub-xeric site and the average stem volume growth is 11 $\mathrm{m}^3\ \mathrm{ha}^{-1}\ \mathrm{yr}^{-1}$. Scots pine and Norway spruce dominated study plots belong to the European forest monitoring network established under the UN-ECE ICP programme (Derome et al., 2002; Merilä et al., 2014).

2.1.3. Norway spruce plot

The Norway spruce dominated stand of this study is conventionally managed forest on herb-rich site, where the latest thinning took place in 1994 and since then the site has been a part of forest monitoring programme. The stand is located in Punkaharju, Finland and the density was approximately 400 stems per ha, the DBH was 28–45 cm, and the height of trees 28–33 m. The average stem volume growth is 8.8 $\mathrm{m}^3\ \mathrm{ha}^{-1}\ \mathrm{yr}^{-1}$ (Merilä et al., 2014).

2.1.4. Mixed-species plots

The two mixed-species plots are located in Sipoo (60°28′N, 25°12′E) and Lapinjärvi (60°39′N, 26°7′E) in Southern Finland. These sites are unmanaged Norway spruce dominated forests (>70% of standing volume), where other tree species were also present. At the time of the scanning in 2014, the stand density was approximately 1300 trees per ha in Sipoo and 1000 trees per ha in Lapinjärvi. For further details, see Rajala et al. (2012).

2.2. Terrestrial laser scanning

The scanning of all forest plots was performed with a RIEGL VZ-400 scanner and a 0.04° resolution. The Silver birch (leaf-off) and Scots pine plots were scanned completely on October 21st, 2014. The scanning of the Norway spruce plot was started on the same day and completed on November 26th, 2014. On both days the weather was cloudy with no rain and light wind, and the temperatures were −1°C and +1°C, respectively. The approximate scanning times were 1, 3 and 4 (2+2) h for the Silver birch, Scots pine and Norway spruce plots, respectively. The Sipoo plot was scanned on November 20th, 2014 and Lapinjärvi plot on November 24th, 2014. On both days the temperature was close to 0 ° C. The number of scanning points per forest plot was selected during the measurements based on visibility in order to cover most of the trees in the scans.

Retroreflectors were attached to tree stems to enable co-registration, which was later performed with the RiScan Pro software. The number of points in the scans, initially and after plot restriction and filtering, were the following: 94 and 35 million for the Silver birch plot, 300 and 58 million for the Scots pine plot, and 340 and 116 million for the Norway spruce plot. For Sipoo

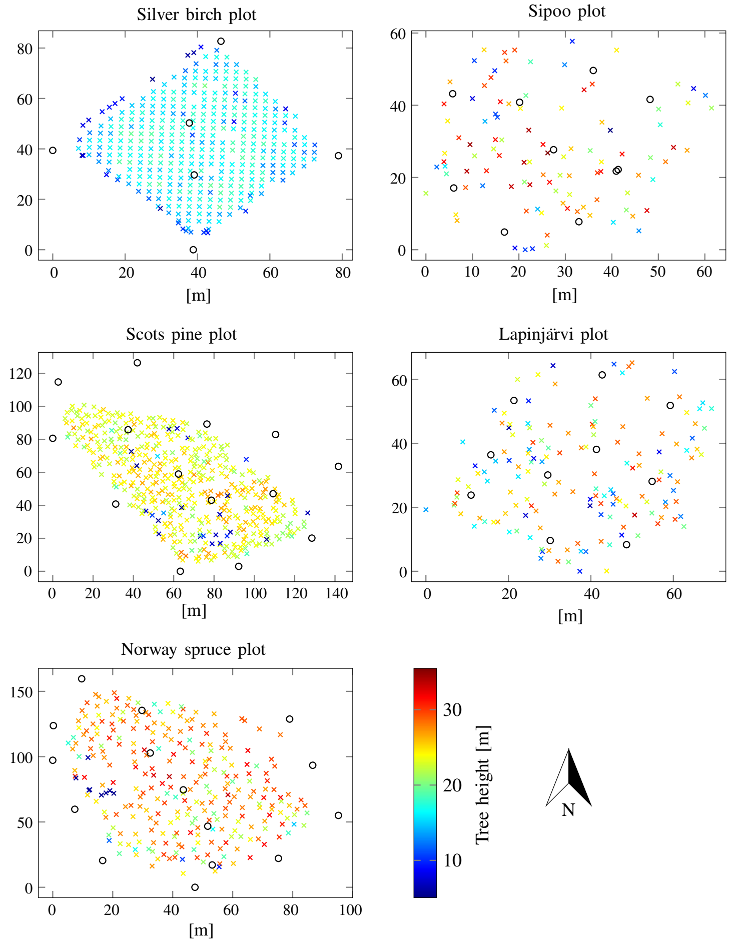


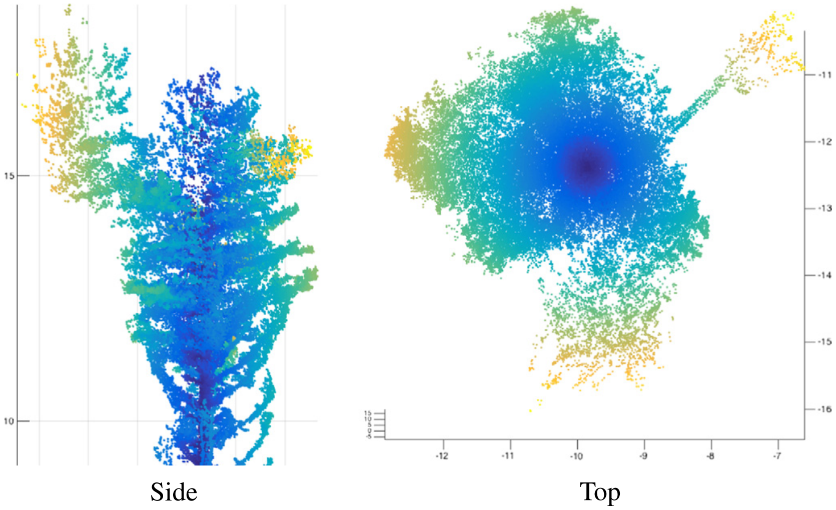
Fig. 1. Forest plot tree location and height map (crosses) and scanner positions (circles). The height colour scale is the same for all plots.

and Lapinjärvi the filtered point counts were 126 and 105 million, respectively. Stem locations and tree heights for each of the study plots are presented in Fig. 1, together with the scanner locations.

2.3. Tree extraction and quantitative structure models

The individual trees were automatically extracted from the point clouds, but only in a suitable region close to the scanners. The extraction method was similar to the one presented in Raumonen et al. (2015) . In the extraction process the point cloud is first filtered for noisy measurements and isolated points that should not contribute to the reconstruction process, and the local ground level is estimated. The filtered point cloud is then partitioned into small subsets or surface patches about 10–20 cm in diameter. These patches are the smallest parts used for tree segmentation. The principal components of the patches are then used to locate the stems based on simple heuristics about the stems being vertical. Next, the located stems are taken as initial sets for the trees and the stems are expanded using surface growing (using the neighbour relation of the patches). There will be separate components that cannot be reached from any tree by surface growing and these components are then connected to the closest point in the closest expanded tree. At this point, most of the trees are uniquely separated except in some cases where the initially expanded stems are connected. The final separation is then achieved with the segmentation of the point cloud into stems and branches. The segmentation follows the procedure presented in Calders et al. (2015b) and Raumonen et al. (2015).

The tree extraction process is prone to some minor errors, and occasional larger errors when some trees are very close to each other and their crowns are occupying the same space. However, the errors in the separation occur mostly in the top parts of the tree crowns, where there is also typically low point cloud coverage due to the extreme angle of the scanner and occlusion. The separation errors most often manifest as a set of branches of a tree incorrectly being contributed as a part of one of its neighbouring tree. An example of this can be seen in Fig. 2. Due to the possibility of such errors, the species classification features were selected to be insensitive in this regard. The features utilize parts of the trees that are expected to be reconstructed most accurately, e.g., branches that originate from the stem. Furthermore, the crown radius, a property that is key to many of the features, is estimated from the volume distribution of the cylinders, rather than their absolute positions, making the estimate more robust should the crown contain parts of the neighbouring trees.



Side

Top

Fig. 2. Example error in tree extraction in on of the Scots pines. Parts of the crown of a neighbouring tree have been contributed to the tree in error.

As a result of the tree extraction, the point cloud is partitioned into subsets that represent single trees. These subset point clouds are used for the reconstruction of QSMs of the individual trees one-by-one. For each tree, the QSM reconstruction process creates another partition of the tree point cloud into small patches, and then segments them into the stem and branches. This new partition utilizes the first segmentation from the tree extraction step to determine a finer and variable-size partition of the tree point clouds. The size of the patches in the partition is determined based on the branching structure and branch size estimates. Next, the patches are segmented into the stem and individual branches similarly as earlier. After the segmentation a cylindrical QSM is reconstructed by fitting cylinders into the segments.

All the QSMs were reconstructed with the same input parameters: the minimum and maximum patch diameters were 3 cm and 12 cm and the relative cylinder length (length/radius) was 5. The parameter values were selected based on previous studies with QSMs to have reasonable models with short computation time (few minutes per tree). Thus the QSMs used in this study were not optimized to be the best possible ones. The optimization of the QSMs could be based on, e.g., median point-to-model distances, as shown in Hackenberg et al. (2015), but that would require even tens of models per tree and thus equivalent increase in computation time. However, one could try a compromise between these two opposites: A reasonable assumption is that the trees and the quality of the measurement data (resolution, noise level, occlusion, etc.) are similar inside the plot. Therefore, at first few trees could be selected for parameter optimization and then use the optimum parameters for all trees.

2.4. Classification methods

In this section, we outline classification methods that are used to classify trees into tree species based on selected *features*, that are defined in Section 2.5. Given M features and K species labels, a training set *TEACH* is a set of pairs ($\mathbf{x}$, *species*), where $\mathbf{x} \in \mathbb{R}^M$ is a list of feature values and $species \in SPECIES = \{species_1, species_2, \ldots, species_K\}$. Either by using the training data, or a model derived from it, the classification methods are able to assign the elements of a test set $TEST = \{\mathbf{x}_i\} \subset \mathbb{R}^M$ into one of the K species classes in *SPECIES*. All the presented methods are also able to present probabilities p_{ij} for an element $\mathbf{x}_i \in TEST$ to belong to a class $species_j \in SPECIES$. Thus, a classification method produces a result set of pairs $(\mathbf{x}_i, \mathbf{p}_i)$, where $\mathbf{x}_i \in TEST$, $\mathbf{p}_i = (p_{i,1}, \ldots, p_{i,K})$, $p_{ij} \in [0, 1]$ and $j \in SPECIES$. The resulting assigned class for a test element $\mathbf{x}_i$ is defined as the class j with the maximum probability p_{ij}.

2.4.1. k-nearest neighbours

The k-nearest neighbours algorithm finds the k elements from the training set *TEACH* that are the closest to a test point $\mathbf{x}_i \in TEST$. The class probability p_{ij} of the element $\mathbf{x}_i$ is defined as the relative number of elements with class j belonging to the subset of k nearest neighbours. Various distance measures can be used when finding the closest elements.

On the implementation level, the `knnsearch` function bundled in MATLAB®was used for this classification method. Number of neighbours parameter k and the distance measure were optimized during computations.

2.4.2. Multinomial regression

For classification based on multinomial regression the `mnrfit` and `mnrval` commands part of the Statistics and Machine Learning Toolbox in MATLAB®were used. In a multinomial logistic regression model, the probability p_{ij} of a sample $\mathbf{x}_i \in TEST$ having class $species_j \in SPECIES$ can be computed as (Agresti, 1990, p. 313):

$$p_{ij} = \frac{\exp\left(\mathbf{x}_i^{\mathrm{T}}\mathbf{b}_k\right)}{\sum\limits_{k=1}^{K} \exp\left(\mathbf{x}_i^{\mathrm{T}}\mathbf{b}_k\right)}. \tag{1}$$

For a selected baseline class (e.g.$j = K$) the coefficients $\mathbf{b}_j$ are set to zero, and for the rest of the classes the coefficients are optimized in an iterative fitting process with the *TEACH*dataset. The predicted outcome class is defined to be the one with the highest probability.

2.4.3. Support vector machine

Support vector machine classification finds a hyperplane that maximizes the margin between the samples of two different classes.

Table 1
List of included tree classification features. Possible units are given in brackets.

	ID	Feature name	Description
Stem branch	1	Stem branch angle	Median of the branching angles of the 1st order branches in degrees. 0 is upwards and 180 downwards. [°]
	2	Stem branch cluster size	Average number of 1st order branches inside a 40 cm height interval for 1st order branches. Each branch can only belong to one interval.
	3	3 Stem branch radius	Mean ratio between the 10 largest 1st order branches measured at the base and the stem radius at respective height.
	4	Stem branch length	Average length of 1st order branches normalized by **DBH**.
	5	Stem branch distance	Average distance between 1st order branches computed using a moving average with a window width 1 m. If window is empty average distance in window is set as half of window width.
Crown	6	Crown start height	Height of first stem branch in tree crown relative to tree height.
	7	Crown height	Vertical distance between the highest and lowest crown cylinder relative to tree height.
	8	Crown evenness	Crown cylinders divided into 8 angular bins. Ratio between extreme minimum heights in bins.
	9	Crown diameter/height	Ratio between crown diameter and height.
Tree	10	DBH/height ratio	Ratio between DBH and total tree height.
	11	DBH/tree volume	Ratio between DBH and total tree volume. [m^{-2}]
	12	DBH/minimum tree radius	Ratio between DBH and the minimum of the vertical bin radius estimates.
	13	Volume below 55% of height	Relative cylinder volume below 55% of tree height.
	14	Cylinder length/tree volume	Ratio between total length of all cylinders and total tree volume. [m^{-2}]
	15	Shedding ratio	The number of branches without children divided by the number of all branches in the bottom third.

In a two-class case the optimization problem can be formulated as a minimization problem as follows:

$$\min_{\mathbf{w},\xi,b} \frac{1}{2}K(\mathbf{w},\mathbf{w}) + C\sum_{i=1}^{l}\xi_i, \quad (2)$$

subject to

$$y_i(K(\mathbf{w},\mathbf{x}_i) - b) \geq 1 - \xi_i \quad (3)$$

$$\xi_i \geq 0, \quad (4)$$

where **w** is the normal vector of the hyperplane and b the plane's location parameter, and C is the penalty parameter. In a two-class case the class indicator y_i of the sample $\mathbf{x}_i$ is 1 if the sample has the class i and -1 otherwise. Furthermore, ξ_i are the slack parameters for a soft margin to allow the overlap of classes. The $K(\cdot,\cdot)$ function is called the kernel function which defines the shape of the class boundary. By using the *linear* kernel function

$$K(\mathbf{a},\mathbf{b}) = \mathbf{a}^T\mathbf{b}, \quad \mathbf{a},\mathbf{b} \in \mathbb{R}^M \quad (5)$$

the boundaries are hyperplanes. Other kernel functions used in this study are *polynomial* and *radial basis function* (RBF), respectively:

$$K(\mathbf{a},\mathbf{b}) = (\gamma\mathbf{a}^T\mathbf{b})^d \quad (6)$$

$$K(\mathbf{a},\mathbf{b}) = e^{-\gamma\|\mathbf{a}-\mathbf{b}\|^2}, \quad (7)$$

where γ and d are kernel parameters. The use of these additional kernel functions allows the decision boundaries to be non-flat. Support vector machine based classification with a linear, polynomial, and RBF kernel functions are noted as SVM_{lin}, SVM_{pol}, and SVM_{rbf}, respectively. For $K > 2$ species classes the classifier can be implemented in $K-1$ steps, where on the kth step the two possible classes are $species_k$ and $SPECIES \setminus species_k$. The `libsvm` package (Chang and Lin, 2011) for MATLAB®was used for the computations. Grid search was used for finding the optimal values for the penalty parameter C and the kernel parameters γ and d.

2.5. Tree features

All the classification methods presented in Section 2.4 are based on feature data. In this study, each feature is a scalar value computed for a single tree (QSM). The features are derived from the geometric and topological properties stored in the reconstructed QSMs. The features were designed to be scale-independent by scaling absolute lengths with, e.g., the tree height or DBH. By using such features the classification should perform well with both young and old trees. Table 1 lists the features that were used in this study. Apart from Features 1, 11 and 14, the features are unitless. Additionally, the Classification features-animation shows how the features are defined and computed. In the animation a single Scots pine model is used to illustrate the features one-by-one (see Supplementary data).

Stem branches (SB) are first-order branches that originate from the stem. The branching angle was computed as the angle between the axes of the first cylinder in the branch and its parent cylinder. The branching cluster size, Feature 2, was approximated by inspecting the start points of the SBs. An inspection height interval with a width of 40 cm was centred at the height of each SB. The number of SBs inside the interval was recorded and the branches were flagged as *used*. Only the SBs without the *used* flag were computed, thus allowing any branch to be part of only one branching cluster; i.e., height interval. At the end, the cluster sizes were averaged for every tree.

DBH was computed by fitting a cylinder at the standard height to the stem point cloud during the reconstruction process and stored in the tree model. The SB radius, Feature 3, was defined to be the average ratio between the radii of the SBs and their respective parent cylinders. Only ten SBs with the largest radius at their base were selected for this feature to get a better separation between species.

Using the topological information stored in a cylinder model the set of crown cylinders was found, using the following algorithm, that is designed to exclude dead branches at the bottom of the stem:

1. Initialize the crown set as cylinders that have a branching order higher than three. If the initial set is empty, the minimum order is lowered until the set becomes non-empty.

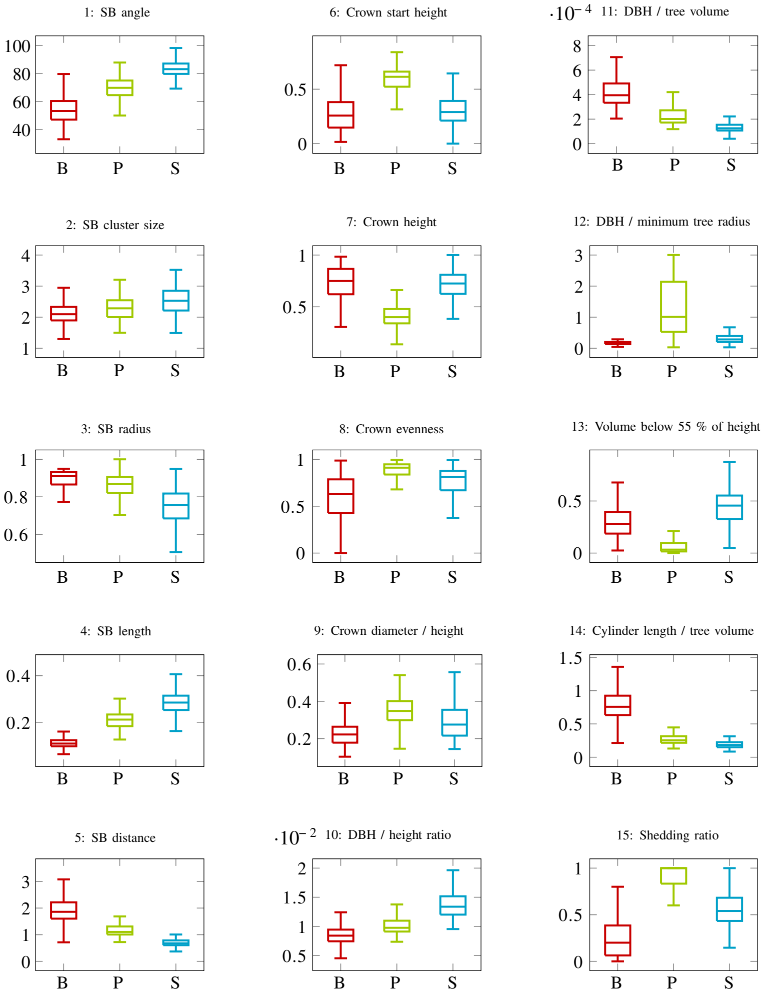


Fig. 3. Box plots of classification features for each tree species: Silver birch (B), Scots pine (P) and Norway spruce (S). The vertical line inside the box is the median. Box limits give the 1st and 3rd quartiles of the distribution and the whiskers extend to 1.5 times the distance between the 1st and 3rd quartiles, or the distribution extremes.

2. As long as the crown set extends, append the parent cylinders of the crown set that are not part of the stem.
3. Append to the crown set cylinders that are not part of the stem but whose start point is higher than the lowest starting point of crown cylinders connected to the stem.
4. As long as the crown set extends, append the child cylinders of the crown set.

The initial crown height is defined as the relative starting height of the lowest SB in the crown. The crown height is the difference between the lowest and highest crown cylinders normalized by the tree height. To analyse how evenly the crown bottom is distributed, the crown set is divided into eight angular bins around the stem, and the minimum vertical point is computed. The crown evenness feature is the ratio between the highest and lowest of these values.

Features 9 and 12 require an estimate of either the tree or the crown radius. Rather than computing the radius estimate directly from the positions of the cylinders furthest from the stem, we use a radial volume distribution for robustness. To estimate the tree radius at different heights, a tree is divided into three vertical bins, and the centre point of each bin is defined as the average of mean points of stem cylinders in the bin. If the bin does not contain stem cylinders the centre of the previous bin is used. The tree radius estimate in a vertical bin is defined as the radius of a cylinder whose axis is vertical and goes through the bin centre point, and which contains 90% of the volume of the cylinders in that bin. The crown diameter is estimated as two times the maximum vertical bin radius.

For Feature 13, the branch cylinder volume distribution is considered in the vertical direction. A good vertical limit 55% of the total tree height was found by testing numerous alternatives. For Feature 15 only the bottom third is considered as it is expected to contain most of the dead/shed branches, which are defined as branches without child branches.

To see the value range and the level of separation between species for the proposed features Fig. 3 visualizes the distribution of each feature per species. The Species separation-animation also shows the

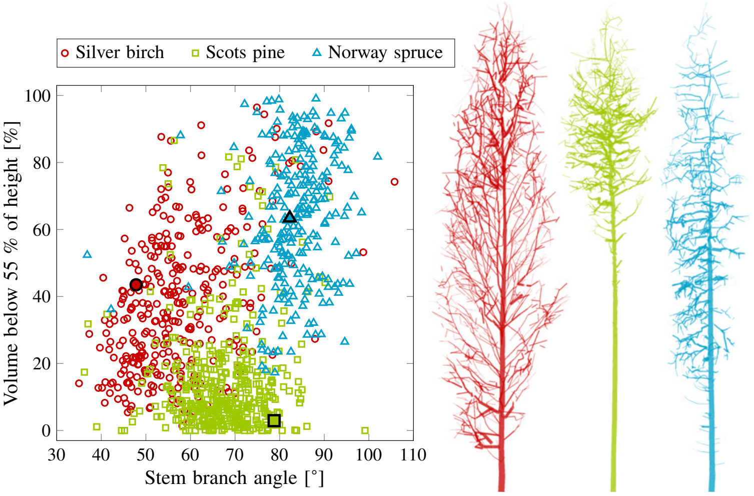


Fig. 4. Example QSMs and feature distribution for Features 1 and 13. The highlighted markers in the distribution correspond to the colour-coded models.

distribution of feature values per species, together with an example model of each species and the respective feature values of these models (see Supplementary data). Furthermore, Fig. 4 visualizes the two-dimensional feature space of Features 1 (stem branch angle) and 13 (volume below 55% of height). Three individual QSMs are also shown and their mapped values in the selected feature space highlighted. The values in both figures were computed using the full population of 358 Silver birch, 457 Scots pine, and 276 Norway spruce trees from the three single-species forest plots.

3. Results

In this section, we first use the three single-species Punkaharju plots for three different purposes: First, we optimize the parameters of the classification methods, then we determine the optimal feature sets, and finally we study the effect of the size of the training dataset. In Section 3.4, we test the classification performance of the optimal parameters and feature sets on mixed-species forest plots, both by using only the single-species plots as training data, and by supplementing the training data with samples from the mixed-species plots.

3.1. Optimal parameters

The parameters of the k-NN method were optimized by using 50 samples of each species and 10-fold cross-validation. The value of k was varied from 2 to 20 and the distance measure in the following parameter set: `euclidean`, `seuclidean`, `cityblock`, `chebychev`, `minkowski`, `mahalanobis`, `cosine`, `correlation`, and `spearman`.

Table 2
Limits and results for SVM kernel parameters γ^* and d, and the penalty parameter C^* grid search.

	C^*	γ^*	d
Minimum	−5	−15	2
Maximum	15	3	5
Increment	1	1	1
Optimums			
Linear	−2	–	–
Polynomial	−5	2	2
RBF	1	−3	–

The highest classification accuracy was achieved with $k = 4$ and the standardized Euclidean distance (`seuclidean`). This combination of parameters was used in all following tests, and is noted as 4-NN.

In order to find the optimal SVM kernel and penalty parameters, a grid search was performed for all of the kernel types. The following convenience parameters were defined for the grid search:

$$C^* = \log_2 C \tag{8}$$

$$\gamma^* = \log_2 \gamma. \tag{9}$$

The grid search was performed on the same 50 samples of each tree species, and was carried out using 5-fold cross-validation provided by the `libsvm` package. The grid limits and results for the search are presented in Table 2. The parameters that yielded the highest classification accuracy were fixed for further classification tests for each kernel type.

Table 3
Classification accuracy $\bar{p}$ in percentages and standard deviation σ for cross-validation in percentage points for best feature combinations with increasing feature count. The highest accuracy for each count is highlighted, and the total maximum value is underlined.

	4-NN		**MNR**		**SVM$_{\mathbf{lin}}$**		**SVM$_{\mathbf{pol}}$**		**SVM$_{\mathbf{rbf}}$**	
Feat. count	$\bar{p}$	σ	$\bar{p}$	σ	$\bar{p}$	σ	$\bar{p}$	σ	$\bar{p}$	σ
1	**85.3**	3.0	76.7	3.0	80.7	3.0	84.6	3.0	81.1	3.0
2	**92.0**	1.4	89.8	2.8	88.8	1.4	90.6	2.8	90.1	2.8
3	**94.6**	1.3	91.5	2.8	91.2	2.6	92.7	2.4	92.3	1.9
4	**95.4**	1.3	92.5	2.1	92.7	2.1	94.7	2.0	93.8	2.0
5	**96.1**	2.2	93.0	2.6	93.6	1.7	95.4	1.7	94.9	1.7
6	**96.2**	1.5	93.6	2.0	93.6	1.7	95.8	2.9	95.1	1.7
7	**96.8**	1.9	93.8	1.2	93.8	1.8	96.2	1.8	95.5	2.0
8	**96.6**	1.8	94.2	2.6	94.1	2.1	96.4	1.8	95.5	1.8
9	**96.8**	1.6	94.2	2.7	94.4	2.4	96.4	2.4	95.8	2.2
10	**96.9**	2.1	94.3	2.1	94.5	2.2	96.7	1.9	95.8	1.8
11	96.6	1.5	94.4	2.3	94.6	1.9	**96.8**	1.9	95.9	1.5
12	**96.7**	1.2	94.5	1.8	94.3	2.2	**96.7**	2.0	95.9	1.8
13	**96.5**	1.8	94.4	2.1	94.2	2.8	96.3	2.2	95.9	2.1
14	**96.4**	1.6	94.6	2.2	94.4	1.7	96.1	1.6	95.8	1.7
15	**96.1**	1.8	93.9	1.8	94.1	1.8	95.5	1.8	95.7	1.8

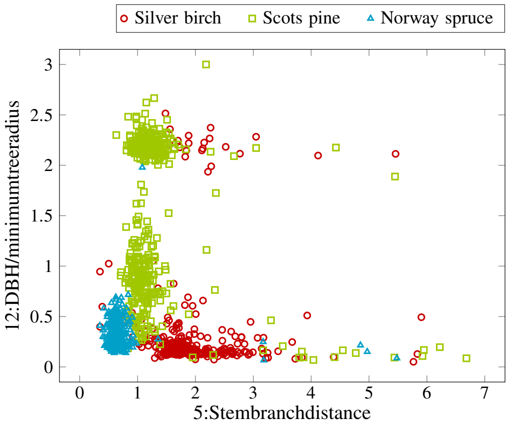


Fig. 5. Data separation with the feature pair that produced the maximum classification accuracy with the 4-NN and $\mathrm{SVM}_{\mathrm{lin}}$ methods.

3.2. Optimal feature sets

To find the optimal classification features, the included feature count was varied from 1 to 15. For each feature count all the possible feature combinations were tested using a 10-fold cross-validation. The remaining trees of each species after the parameter optimization were divided into ten subsets of equal size. Each subset was then classified by using the remaining nine subsets of each species as training data. The 10-fold cross validation was selected to show the classification performance in close to optimal conditions, where the amount of the training data far exceeds the amount of the test data. The maximum classification accuracies for each method and feature count are listed in Table 3.

When using just one feature, the maximum classification accuracy, 85.3%, was achieved with the 4-NN method and Feature 5: Stem branch distance. This combination was able to separate Silver birches and Norway spruce trees from each other but mixed birches with Scots pines, as 12.4% of birches were misidentified as pines and 15.1% *vice versa*.

When the feature count was two, the 4-NN and $\mathrm{SVM}_{\mathrm{lin}}$ classification methods gave the maximum accuracy with Features 5 and 12. The values of these features are visualized in Fig. 5 to show the high level of species separation. For the remaining three methods, the best combination was Features 11 and 15. The maximum total accuracy, 92.0%, was received with the 4-NN method. With this configuration the standard deviation over the folds was 1.4 pp. and the biggest confusion was again between birches and pines.

For feature counts higher than two there is no clear difference between the methods as the standard deviation of the classification accuracies over the methods remain between 1.23 and 2.89 percentage points. With feature counts from 6 to 15 all the methods result in accuracies above 93.0%.

The highest accuracy, 96.9%, was obtained with the 4-NN method and 10 features. The resulting relative and absolute confusion tables are presented in Table 4. The results show that out of 1010 trees only 31 were misclassified when using this setup.

For further testing, *top feature sets* were selected using the following two criteria: 1) sets whose average classification accuracy over the methods was above 95%; 2) sets whose minimum classification accuracy over the methods was above 94%. 120 combinations fulfilled the first condition, and 16 combinations the second. Out of these combinations, ten fulfilled both and these combinations are listed in Table 5.

Fig. 6 shows the frequency at which each feature is part of the 126 top feature sets. Out of the 15 features, four (1, 10, 13, 14) are part of all the top feature sets. Next, the top feature sets are used for testing the effect of training data size.

The independence level of the classification features was tested by studying the covariance of the features over the total population of 1010 trees. The correlation between Features 6 and 7 is over 99% because none of the included species have crowns that extend below the lowest connecting stem branch. Furthermore, Features 4 and 11 have a correlation over 80%. The connection here is not that obvious as Feature 4 measures stem branch length and Feature 11 total volume of the tree. The correlation is expected to drop for both feature pairs when additional species with varying geometry are included. For the purpose of this study the high correlation between these pairs did not affect the classification accuracies, as is evident from the similar results with feature combinations containing and not containing the highly correlated feature pairs, e.g., in Table 5.

3.3. Effect of the size of training dataset

In order to test how each of the proposed classification methods performs in real applications, the effect of the amount of training data was studied. The number of trees per species in the training dataset was varied from 4 to 150, while the same number for the testing data remained constant at 50. The tree samples were selected randomly from the complete tree population, but the training and testing sets remained disjoint. Sampling was repeated 10 times for each training data size. The optimal parameters from Section 3.1 were used in the test, as well as, the top feature sets from Table 5.

Table 6 shows the total classification accuracy of each method as a function of the amount of training data. The accuracy of a method is averaged over the random repeats and the feature combinations. Furthermore, Fig. 7 shows the average, minimum and maximum classification accuracies for the same data.

The 4-NN method has the highest average classification accuracy and the lowest standard deviation with all training data sizes. With data sizes above or equal to 30, the average classification accuracy remains above 93%, and the minimum accuracy above 88%. The MNR and $\mathrm{SVM}_{\mathrm{lin}}$ methods also perform well with an average accuracy above 91% with 50 samples or more. The standard deviation is a little greater than with the 4-NN method but the minimum classification accuracies still remain above 82%, and with a high number of samples (≥100) even comes near 87%.

The $\mathrm{SVM}_{\mathrm{pol}}$ and $\mathrm{SVM}_{\mathrm{rbf}}$ methods have relatively hight standard deviation with all sample sizes but still below 4 pp. The average accuracy on the $\mathrm{SVM}_{\mathrm{pol}}$ method is very good, but some of top feature sets give poor results dropping the minimum performance quite low, even with a high number of training samples. Similarly, the $\mathrm{SVM}_{\mathrm{rbf}}$ method has a relatively low minimum performance, but unlike with the $\mathrm{SVM}_{\mathrm{pol}}$ method, the effect seems to diminish with the increasing number of samples. For operational use the 4-NN, MNR and $\mathrm{SVM}_{\mathrm{lin}}$ methods seem to provide the most consistent results.

While performing the accuracy testing described above, the correlation between tree properties, the measurement setup, and the probability of an incorrect classification was studied. As the tree sets

Table 4
Relative and absolute confusion tables for 4-NN method and 10 features with highest classification accuracy.

		Predicted					
		B	P	S	B	P	S
	Silver birch	**98.2**	1.5	0.3	**324**	5	1
Correct	Scots pine	3.0	**96.3**	0.7	13	**414**	3
	Norway spruce	1.6	2.0	**96.4**	4	5	**241**
		Relative			Absolute		

Table 5
Top feature sets that fulfilled both conditions with their classification accuracy per method and averaged in percentages. The maximum accuracy over the listed feature sets is highlighted for each method.

#	Feature set	4-NN	MNR	SVM_{lin}	SVM_{pol}	SVM_{rbf}	Average
1	1, 2, 3, 5, 7,10,11,13,14,15	96.2	94.1	94.2	96.1	95.5	95.2
2	1, 2, 3, 5, 8,10,11,12,13,14	96.2	94.3	**94.5**	**96.2**	95.4	95.3
3	1, 2, 3, 6, 8,10,11,13,14,15	96.1	94.3	94.3	95.2	95.5	95.1
4	1, 2, 3, 7, 8,10,11,13,14,15	96.0	94.3	94.3	95.2	95.4	95.0
5	1, 2, 3, 5, 6, 8,10,11,12,13,14	**96.4**	94.1	94.1	96.0	95.3	95.2
6	1, 2, 3, 5, 6, 7, 8,10,11,13,14,15	95.8	94.2	94.1	95.8	95.8	95.1
7	1, 2, 3, 5, 6, 7, 8, 9,10,11,13,14,15	96.1	94.4	94.1	95.8	**95.9**	95.3
8	1, 2, 3, 5, 6, 7, 9,10,11,12,13,14,15	96.0	94.3	94.2	95.5	95.7	95.1
9	1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,13,14,15	96.2	94.2	94.4	95.6	95.8	95.2
10	1, 2, 3, 5, 6, 7, 8, 9,10,11,12,13,14,15	96.2	**94.6**	94.1	95.3	95.7	95.2

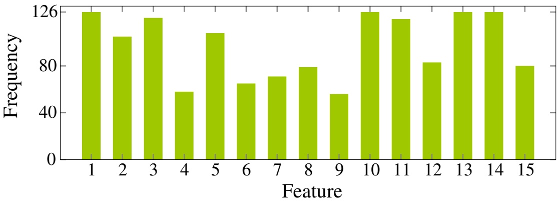


Fig. 6. Number of times a feature was part of the 126 top feature sets.

were randomly sampled, the number of picks per tree was stored, as was the number of incorrect classifications. The probability of incorrect classification was computed as the ratio between the two numbers. The results showed that there was no correlation between the probability to be incorrectly classified and the minimum distance to the laser scanner position. Furthermore, there was no correlation between the probability and estimated tree radius, tree height, and average distance to the laser scanner position.

3.4. Performance on a mixed species forest plots

Above, we used only single-species plots for training and testing. Arguably one can ask how well these results generalize to mixed-species forest, which may allow more variability for tree structures. We only have limited data for this kind of testing, and therefore, we did not use it in the above analysis. To show preliminary results we used two mixed-species plots located in Sipoo and Lapinjärvi. The forests are dominated by Norway spruce trees, but also contains Silver birches and Scots pines among other species. However, as up-to-date tree maps were not available, we manually classified the trees by inspecting point clouds of individual trees and by referencing field-measured tree maps from 2009. We identified 22 Silver birches, 13 Scots pines and 214 Norway spruce trees which were then used for testing.

Training data was sampled randomly from the Punkaharju plots, 100 trees per species. Classification accuracy was tested with all the classification methods with the optimal parameter values and the best feature combinations determined for the three Punkaharju plots listed in Section 3.2. Additionally, training data sampling was repeated 100 times. An additional test was performed where 100 of the Norway spruce trees from the mixed-species plots were used for training and the remaining 114 were tested, while other parameters remained the same. The other two species did not have enough samples to split them into training and testing sets. Table 7 lists the average and maximum classification results for each of the methods per tree species as well as a total accuracy for both training data sources.

When using training data from only the single-species plots the average classification accuracy is between 50% and 55% and the maximum around 70%. The best result, 76.3%, is produced with MNR method. When including Norway spruce tree samples from the mixed-species plot in the training data, all of the average and maximum total accuracies improve. The total average accuracies are still between 50% and 70%, but for the Norway spruce trees, that have sufficient samples from both types of forests, the average classification accuracy is over 80% and the maximum over 90% for some

Table 6
Average classification accuracy and standard deviation as a function of training data size per species. The average in percentages and the std in percentage points are computed over the 126 top feature sets and the 10 repeats with random data.

	4-NN		MNR		SVM_{lin}		SVM_{pol}		SVM_{rbf}	
Sample size	$\bar{p}$	σ	$\bar{p}$	σ	$\bar{p}$	σ	$\bar{p}$	σ	$\bar{p}$	σ
4	87.8	7.4	55.1	17.1	60.4	25.0	68.7	19.1	43.5	24.1
10	92.1	2.4	88.0	4.6	87.6	4.0	88.2	3.3	71.6	11.6
20	92.9	1.4	89.1	2.8	90.1	2.2	89.6	2.4	82.5	3.7
30	93.0	1.4	90.2	2.6	90.1	2.2	89.7	2.4	85.9	2.9
40	93.3	1.4	90.9	2.4	91.1	2.2	90.7	2.8	86.8	3.0
50	93.7	1.5	91.1	2.1	91.5	2.1	91.0	3.0	87.2	3.6
60	94.2	1.5	91.5	1.9	91.7	1.9	91.4	2.8	88.1	3.4
70	94.1	1.5	91.5	1.6	91.7	2.0	91.2	3.4	88.3	2.9
80	94.1	1.4	91.7	1.6	91.7	1.9	91.9	3.6	88.6	2.9
90	94.0	1.1	91.4	1.7	91.4	1.7	92.0	3.4	89.2	3.3
100	94.5	1.3	91.3	1.6	91.6	2.0	92.4	3.1	89.8	2.9
110	94.6	1.2	91.6	1.4	91.8	1.8	92.7	3.2	89.4	2.8
120	94.6	1.3	91.7	1.5	91.7	2.0	92.8	3.4	89.6	2.7
130	94.5	1.2	91.8	1.6	92.0	1.8	93.2	2.8	89.9	2.9
140	94.7	1.3	91.9	1.7	92.0	1.9	93.2	3.2	89.7	2.8
150	94.6	1.2	91.9	1.7	92.1	1.8	93.3	3.4	90.4	2.2

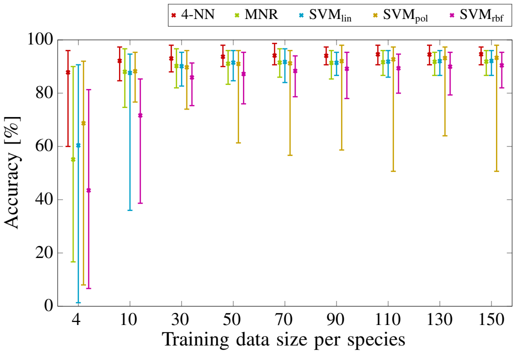


Fig. 7. Average, minimum and maximum classification accuracy with changing training data size per species.

of the methods. For the other two species, which don't have training data from the mixed-species forests, the average and maximum accuracies either stay the same or decrease.

Fig. 8 illustrates values for two of the classification features for the trees from both the single-species and mixed-species forest plots. As the number of samples for two of the species is so low it is hard to make definite conclusions, but at least for the two visualized dimensions the species separation seems lower. Especially, mixed-species plot Silver birches overlap with single-species plot Scots pines and mixed-species plot pines overlap with both mixed and single-species plot Norway spruce trees.

4. Discussion

The feature combination test with the three Punkaharju single-species plots showed that, with comprehensive training data, it is possible to find a feature combination for each of the classification methods such that the average classification accuracy is over 93%. However, when the size of the training dataset was more limited, as it would likely be in real applications, the differences in the classification methods started to show. The fact that with only 30 samples per species, the *minimum* accuracy for the SVM$_{\text{lin}}$ method was above 83%, shows promise for real applications; Sufficient training data can be obtained from the same forest through manual classification. Alternatively, if applicable training data are already available, a complete forest can be classified in a fully automatic procedure, where the classification of a single tree takes only a fraction of a second after the QSM has been computed.

Table 7
Average and maximum classification accuracies for each classification method for the mixed species forest plot in percentages. First half shows results with single-species plot training data and the second half with a combination of single and mixed-species plot training data. The highest maximum total classification accuracies for both types of training data have been highlighted in green, and the maximum accuracies for the Norway spruce trees have been highlighted in black.

Training data	**Method**	Average				Maximum			
		Total	Birch	Pine	Spruce	Total	Birch	Pine	Spruce
Single	4-NN	48.9	18.7	39.8	52.6	61.0	36.4	61.5	67.3
	MNR	54.6	41.5	58.9	55.7	**76.3**	72.7	92.3	80.4
	SVM$_{\text{lin}}$	52.9	29.5	42.9	55.9	64.7	50.0	61.5	71.5
	SVM$_{\text{pol}}$	55.8	19.0	50.0	59.9	73.1	36.4	84.6	79.9
	SVM$_{\text{rbf}}$	55.1	24.5	43.1	59.0	67.5	40.9	84.6	73.4
Single + Mixed	4-NN	52.7	18.3	33.2	61.6	67.1	36.4	53.8	**79.8**
	MNR	70.3	39.2	34.1	80.4	**81.2**	68.2	53.8	**93.0**
	SVM$_{\text{lin}}$	66.3	29.2	28.5	77.8	74.5	50.0	38.5	**88.6**
	SVM$_{\text{pol}}$	68.8	18.9	33.2	82.5	75.2	27.3	46.2	**92.1**
	SVM$_{\text{rbf}}$	68.2	24.3	29.5	81.1	73.8	40.9	38.5	**89.5**

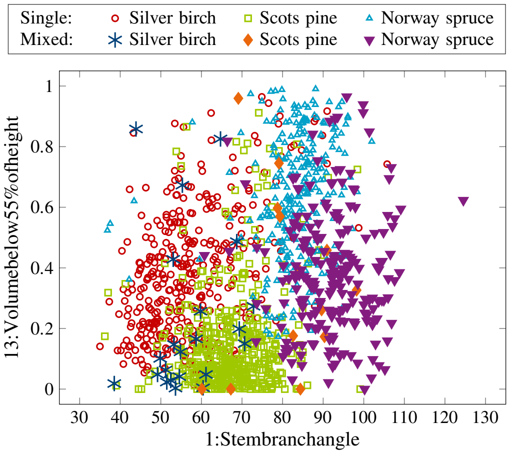


Fig. 8. Mixed-species plot species separation with two classification features.

A preliminary mixed-species plot classification test was performed on a limited dataset. When the training data only included samples from single-species forests and neither the classification method parameters or the feature combinations were optimized, the classification accuracies remained low. However, by including samples of Norway spruce trees from both the mixed and single-species plots in the training data the classification accuracy improved significantly, especially for the Norway spruce species. The result suggests that accurate species classification is possible also in mixed-species forests when adequate training data are available. However, it is clear that further testing is needed to determine the classification performance in a mixed-species forest, when suitable data become available.

Even though the tree separation was not perfect as even coarse errors did occur, all of the classification methods were able to classify trees with an accuracy over 94%, with some feature combinations in the cross-validation test. This shows that the selected features are robust in terms of the tree separation, and that the classification methods are suitable and robust for this classification problem. Furthermore, three of the methods, 4-NN, MNR and SVM$_{\text{lin}}$, performed very well and gave consistent results when studying the effect of the size of the training dataset, which shows that QSMs contain sufficient species-specific characteristics for classification.

Three species were considered in this study, but the computation of the proposed tree features and the use of the classification methods should generalize well to a larger number of species. In the future, the tests should be repeated with a larger number of species from varying types of forests to see, whether the accuracy also generalizes. It is well known that the top parts of tall trees are poorly covered in TLS measurements, especially in dense environments. Naturally, QSMs inherit this shortcoming, and thus if the differences between two tree species are mainly focused on the top parts of the trees, the proposed classification method might fail to separate them. A further study could also include testing for the effect of leaves on the classification accuracy. From previous studies we know that the presence of leaves will decrease the quality of the QSM, but the effect on species classification still remains unknown.

The QSMs used in this paper were not optimized, but a reasonable set of input parameters was used for all trees. A further study on the effects of the input parameters on the classification should be carried out. One would expect that more accurate QSMs will improve the classification accuracy. The QSMs used in this study consisted of circular cylinders, but similar models can be computed with other, more complicated elements as well (Åkerblom et al., 2015). The use of more complex shapes might give access to even more tree features, such as cross-section shapes, but can decrease the reconstruction accuracy and stability. In any case, the presented results show even the relatively simple circular cylinder QSMs can produce high classification accuracies.

The presented 15 classification features yielded good results, but the use of QSMs allows the computation of numerous other features as well. It is also possible to combine the features computed from QSM with features computed directly from the spatial point cloud, or hyperspectral data, to achieve even better separation between species. However, in comparison to previous studies (Puttonen et al., 2011, 2010), the presented QSM based approach produced similar or even higher classification accuracies without using hyperspectral data or even TLS intensity values. Furthermore, the proposed classification method is fully automatic and works on a lower resolution scanning data than the bark texture based method proposed by Othmani et al. (2013).

Optimization can also be done by increasing the number of scan positions, or, e.g., by elevating the scanner to get better coverage in higher tree parts. The improvement in the scan detail will translate to the quality of the reconstructed models. Alternatively, it could be studied how high classification accuracies can be achieved by using a more coarse scanning setup. It might be possible to integrate automatic species identification to forest inventories done with TLS if the classification features are chosen correctly.

5. Conclusion

We have presented a novel, fully automatic tree species classification approach for terrestrial laser scanning data. The approach is based on reconstructing quantitative structure models (QSM) of the trees, which enables the computation of tree properties that have not been available before for species classification. 15 classification features that utilize the geometry and topology stored in the QSMs were proposed, and their suitability for separating tree species was studied in various tests using three different classification methods: 4 nearest neighbours (4-NN), multinomial regression (MNR) and support vector machines. For the latter, three different kernel functions were also considered: linear (SVM_{lin}), polynomial (SVM_{pol}), and radial basis function (SVM_{rbf}).

The classification accuracy was tested on tree models reconstructed from three single-species forest plots. Over 1000 trees were used in a 10-fold cross-validation classification test with all classification methods. Furthermore, all possible feature combinations were tested, and the best classification accuracy, 96.9%, was achieved with the 4-NN method and 10 features. *With the feature count between 6 and 15, it was possible to find a feature combination that resulted in an average classification accuracy above 93% for all the methods.*

To further test the performance of the classification methods, 126 of the top feature sets were selected to test how the amount of training data affected the classification accuracy. The training and testing trees were selected randomly from the total tree population, and the process was repeated ten times for each feature combination and classification method. The results showed that 4-NN, MNR and SVM_{lin} gave consistent results with all the sets, and were able to classify trees with an average accuracy above 90% and minimum accuracy over 82%, with the training sample number per species greater or equal to 30. The SVM_{pol} and SVM_{rbf} methods had lower minimum accuracies as they did not perform well with all of the top feature sets. Thus, more caution should be exercised when selecting feature sets for these particular methods.

Preliminary testing on mixed-species forest plot trees showed that training data collected solely from single-species forest plots is not sufficient for good results. Although, more testing is required when more comprehensive training data becomes available, adding training data from mixed-species forest plots even for just one species improved the classification accuracy to be over 80% for the MNR method.

Our study showed that tree features made accessible through QSM reconstruction can outperform the existing tree species classification approaches based only on 3D spatial, or hyperspectral, point cloud data. QSMs provide access to features based on detailed geometry and branching structure making the source data more than three-dimensional. The study also showed that QSMs provide a very robust basis for classification with little need for tuning. Three different methods could accurately and with small variation classify the trees based on QSMs that were not optimized for each tree and with considerable errors in the tree separation. Finally, the proposed species recognition approach is fully automatic because the required pre-processing steps of tree separation of the underlying forest plot (Raumonen et al., 2015) and the QSM reconstruction are automatic.

Acknowledgments

The authors would like to thank Esko Oksa for technical support and arrangements of the TLS, Nina Heiska and Hannu Heinonen (Nordic Geo Center Oy) for the TLS, and Päivi Merilä and Matti Rousi for a permission to use their long-term monitoring plots in this study.

This study was funded by the Academy of Finlandresearch projects Centre of Excellence in inverse problems[284715], and Challenges of forest management in the changing environment[292930]. Data from multi-species stands was collected as part of the project Resource utilization pattern of wood-inhabiting fungi [292899].

Appendix A. Supplementary data

Supplementary data to this article can be found online at http://dx.doi.org/10.1016/j.rse.2016.12.002.

References

Agresti, A., 1990. Categorical Data Analysis. Wiley Series in Probability and StatisticsWiley.

Åkerblom, M., Raumonen, P., Kaasalainen, M., Casella, E., 2015. Analysis of geometric primitives in quantitative structure models of tree stems. Remote Sens. 7 (4), 4581–4603.

Calders, K., Disney, M., Nightingale, J., Origo, N., Barker, A., Raumonen, P.A., Lewis, P., Burt, A., Brennan, J., Fox, N., 2015a. Traceability of essential climate variables through forest stand reconstruction with terrestrial laser scanning. Proceedings of SilviLaser 2015. pp. 122–124.

Calders, K., Newnham, G., Burt, A., Murphy, S., Raumonen, P., Herold, M., Culvenor, D., Avitabile, V., Disney, M., Armston, J., Kaasalainen, M., 2015b. Nondestructive estimates of above-ground biomass using terrestrial laser scanning. Methods Ecol. Evol. 6 (2), 198–208.

Chang, C.C., Lin, C.J., 2011. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm.

Derome, J., Lindgren, M., Merilä, P., Beuker, E., Nöjd, P., 2002. Forest condition monitoring under the UN/ECE and EU programmes in Finland. Forest Condition Monitoring in Finland — National Report 2005, 11–20.

Haala, N., Reulke, R., Thies, M., Aschoff, T., 2004. Combination of terrestrial laser scanning with high resolution panoramic images for investigations in forest applications and tree species recognition. ISPRS working Group 1,

Hackenberg, J., Spiecker, H., Calders, K., Disney, M., Raumonen, P., 2015. SimpleTree—an efficient open source tool to build tree models from TLS clouds. Forests 6 (11), 4245–4294.

Holopainen, M., Kankare, V., Vastaranta, M., Liang, X., Lin, Y., Vaaja, M., Yu, Xiaowei, Hyyppä, J., Hyyppä, H., Kaartinen, H., Kukko, A., Tanhuanpää, T., Alho, P., 2013. Tree mapping using airborne, terrestrial and mobile laser scanning — a case study in a heterogeneous urban forest. Urban For. Urban Green. 12 (4), 546–553.

Lin, Y., Herold, M., 2016. Tree species classification based on explicit tree structure feature parameters derived from static terrestrial laser scanning data. Agric. For. Meteorol. 216, 105–114.

Meier, I.C., Leuschner, C., Marini, E., Fender, A.C., 2016. Species-specific effects of temperate trees on greenhouse gas exchange of forest soil are diminished by drought. Soil Biol. Biochem. 95, 122–134.

Merilä, P., Mustajärvi, K., Helmisaari, H.S., Hilli, S., Lindroos, A.J., Nieminen, T.M., Nöjd, P., Rautio, P., Salemaa, M., Ukonmaanaho, L., 2014. Above-and below-ground N stocks in coniferous boreal forests in Finland: implications for sustainability of more intensive biomass utilization. For. Ecol. Manag. 311, 17–28.

Othmani, A., Lew Yan Voon, L., Stolz, C., Piboule, A., 2013. Single tree species classification from terrestrial laser scanning data for forest inventory. Pattern Recogn. Lett. 34 (16), 2144–2150.

Possen, B.J., Anttonen, M.J., Oksanen, E., Rousi, M., Heinonen, J., Kostiainen, K., Kontunen-Soppela, S., Heiskanen, J., Vapaavuori, E.M., 2014. Variation in 13 leaf morphological and physiological traits within a silver birch (*Betula pendula*) stand and their relation to growth. Can. J. For. Res. 44 (6), 657–665.

Puttonen, E., Jaakkola, A., Litkey, P., Hyyppä, J., 2011. Tree classification with fused mobile laser scanning and hyperspectral data. Sensors 11 (5), 5158–5182.

Puttonen, E., Suomalainen, J., Hakala, T., Räikkönen, E., Kaartinen, H., Kaasalainen, S., Litkey, P., 2010. Tree species classification from fused active hyperspectral reflectance and LIDAR measurements. For. Ecol. Manag. 260 (10), 1843–1852.

Rajala, T., Peltoniemi, M., Pennanen, T., Mäkipää, R., 2012. Fungal community dynamics in relation to substrate quality of decaying Norway spruce (*Picea abies* [L.] Karst.) logs in boreal forests. FEMS Microbiol. Ecol. 81 (2), 494–505.

Raumonen, P., Casella, E., Calders, K., Murphy, S., Åkerblom, M., Kaasalainen, M., 2015. Massive-scale tree modelling from TLS data. ISPRS Ann. Photogramm., Int. Arch. Photogramm. Remote. Sens. Spat. Inf. Sci. II-3/W4, 189–196.

Raumonen, P., Kaasalainen, M., Åkerblom, M., Kaasalainen, S., Kaartinen, H., Vastaranta, M., Holopainen, M., Disney, M., Lewis, P., 2013. Fast automatic precision tree models from terrestrial laser scanner data. Remote Sens. 5 (2), 491–520.

Vauhkonen, J., Hakala, T., Suomalainen, J., Kaasalainen, S., Nevalainen, O., Vastaranta, M., Holopainen, M., Hyyppä, J., 2013. Classification of spruce and pine trees using active hyperspectral LiDAR. IEEE Geosci. Remote Sens. Lett. 10 (5), 1138–1141.

Article V

Non-Intersecting leaf insertion algorithm for tree structure models

Åkerblom, M., Raumonen, P., Casella, E., Disney, M.,
Danson, F.M., Gaulton, R., Schofield, L.A., and Kaasalainen, M.

Interface Focus, 8(2).

DOI: http://dx.doi.org/10.1098/rsfs.2017.0045

2018

INTERFACE FOCUS

rsfs.royalsocietypublishing.org

Research

Cite this article: Åkerblom M, Raumonen P, Casella E, Disney M, Danson FM, Gaulton R, Schofield LA, Kaasalainen M. 2018 Non-intersecting leaf insertion algorithm for tree structure models. *Interface Focus* **8**: 20170045. http://dx.doi.org/10.1098/rsfs.2017.0045

Accepted: 7 December 2017

One contribution of 12 to a theme issue 'The terrestrial laser scanning revolution in forest ecology'.

Subject Areas:
biomathematics, computational biology, mathematical physics

Keywords:
leaf insertion, leaf distribution, quantitative structure model, laser scanning, tree reconstruction

Author for correspondence:
Markku Åkerblom
e-mail: markku.akerblom@tut.fi

Non-intersecting leaf insertion algorithm for tree structure models

Markku Åkerblom[1], Pasi Raumonen[1], Eric Casella[2], Mathias Disney[3,4], F. Mark Danson[5], Rachel Gaulton[6], Lucy A. Schofield[7] and Mikko Kaasalainen[1]

[1]Laboratory of Mathematics, Tampere University of Technology, PO Box 553, 33101 Tampere, Finland
[2]Centre for Sustainable Forestry and Climate Change, Forest Research, Farnham GU10 4LH, UK
[3]Department of Geography, University College London, Gower Street, London WC1E 6BT, UK
[4]NERC National Centre for Earth Observation (NCEO), UK
[5]School of Environment and Life Sciences, University of Salford, Salford M5 4WT, UK
[6]School of Engineering, Newcastle University, Newcastle upon Tyne NE1 7RU, UK
[7]School of Humanities, Religion and Philosophy, York St John University, York YO31 7EX, UK

MÅ, 0000-0002-6512-232X; PR, 0000-0001-5471-0970; EC, 0000-0002-5429-7159; MD, 0000-0002-2407-4026; FMD, 0000-0002-3984-0432; RG, 0000-0002-0706-0298

We present an algorithm and an implementation to insert broadleaves or needleleaves into a quantitative structure model according to an arbitrary distribution, and a data structure to store the required information efficiently. A structure model contains the geometry and branching structure of a tree. The purpose of this work is to offer a tool for making more realistic simulations of tree models with leaves, particularly for tree models developed from terrestrial laser scanning (TLS) measurements. We demonstrate leaf insertion using cylinder-based structure models, but the associated software implementation is written in a way that enables the easy use of other types of structure models. Distributions controlling leaf location, size and angles as well as the shape of individual leaves are user definable, allowing any type of distribution. The leaf generation process consist of two stages, the first of which generates individual leaf geometry following the input distributions, while in the other stage intersections are prevented by carrying out transformations when required. Initial testing was carried out on English oak trees to demonstrate the approach and to assess the required computational resources. Depending on the size and complexity of the tree, leaf generation takes between 6 and 18 min. Various leaf area density distributions were defined, and the resulting leaf covers were compared with manual leaf harvesting measurements. The results are not conclusive, but they show great potential for the method. In the future, if our method is demonstrated to work well for TLS data from multiple tree types, the approach is likely to be very useful for three-dimensional structure and radiative transfer simulation applications, including remote sensing, ecology and forestry, among others.

1. Introduction

Leaves and needles are essential for the functioning of plants and their interaction with the environment. They are also the main part of the vegetation interacting with remote sensing measurements. Thus, the ability to measure and model leaf distributions of plants has great importance and many applications in ecology, forest research and remote sensing [1–3].

We will present an algorithm to generate leaf cover on any plant structure model with any underlying distribution for the leaf parameters. Although the process could be used with any type of plant, this article focuses only on trees. The leaf parameter distributions are supported by quantitative structure models (QSMs) of trees, and the generated leaves are non-intersecting. This allows, among other things, the use of more realistic leaf distributions in

THE ROYAL SOCIETY
PUBLISHING

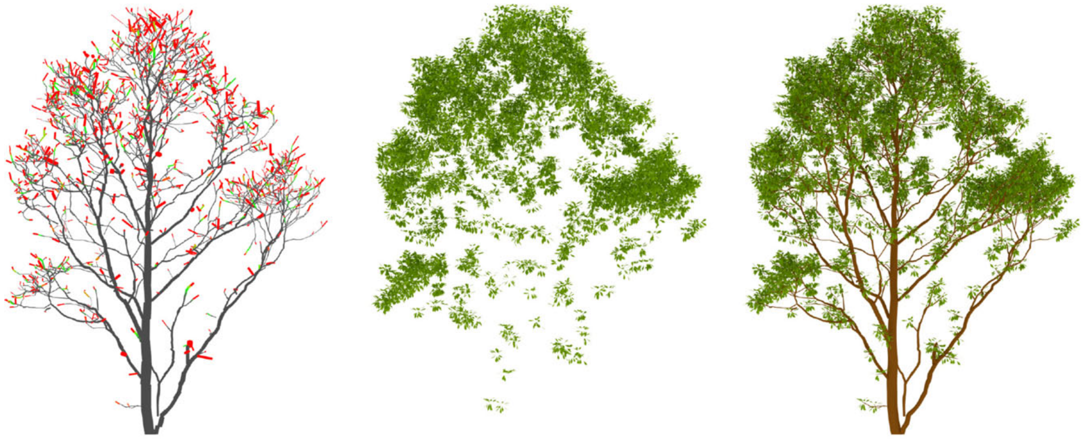


Figure 1. A QSM supports a leaf area distribution (grey: no leaves; green: some leaves; red: a lot of leaves), which can be sampled to generate non-intersecting leaves and inserted into the structure model.

gap fraction- and radiative transfer-based simulations, in comparison with the previously suggested uniform layers of possibly intersecting leaves [4].

The above-ground biomass of a tree consists mainly of leaves, and woody material in the trunk and branches. In recent years, various methods have been presented to reconstruct the woody parts of a tree in a quantitative manner from terrestrial laser scanning (TLS) data [5,6]. Furthermore, it is possible to estimate foliage distribution from similar data [7] (for further information, see [8]). However, reconstructing both the woody and leaf parts at the same time is more challenging due to self-occlusion effects, and the complex nature of leaf–wood separation from TLS data, which has been studied extensively [9,10].

An alternative to *extracting* the leaves from TLS data is scanning the tree during the leaf-off season, and then trying to *insert* leaves after reconstructing the woody structure. To generate a leaf cover that is statistically similar to the original, certain leaf property distributions have to be estimated [11]. Such approaches do not aim to reconstruct real leaves but rather the underlying leaf distribution, which can be sampled to produce leaf covers that are statistically similar to the real one. The approach is limited to deciduous, broadleaf canopies. However, from this we may learn how to improve and develop methods for separation and re-insertion of green material in evergreen broadleaf and needleleaf trees.

Measuring leaf position, size and orientation by hand is extremely laborious [11] as one can have millions of leaves per tree. Great progress in measurement systems and data analysis has meant that remote sensing can now be used to detect leaf properties. Methods have been presented to estimate the three-dimensional distribution of leaf material from TLS data [7,12]. Furthermore, methods for measuring leaf orientation distribution (LOD) from similar data have been presented in [13] and more recently in [14]. Determining leaf size distribution (LSD) remotely is more challenging as it requires the detection of leaf edges [15], which is also challenging due to the decrease in data point density higher in the canopies, when scanning from the ground. However, sampling leaf size by hand is faster and less error prone than leaf angle, especially when carried out in a destructive manner.

The algorithm we present in this paper populates a QSM of the woody parts of a tree with leaves, resulting in a model with inserted leaves (L-QSM). The algorithm generates leaves based on user-defined leaf property distributions that may be estimated with the methods presented above, or alternatively by using distributions parametrized by branch properties such as branch order. The basic steps of the procedure are illustrated in figure 1, which shows an example leaf area distribution supported by a QSM, leaves generated by sampling the distribution and the final product, which is an L-QSM.

The algorithm is designed to work with models consisting of any type of geometry, but we use models that are a collection of cylinders, i.e. cylindrical QSMs [5]. The leaf insertion procedure works on *blocks*, which is essentially the largest unit of the structure model that can be assumed to have uniform leaf distribution parameters that can define, for example, limits for the number of leaves, leaf size and orientation. Because certain tree species can have a different leaf density along branches, the blocks can be smaller than the branch. Thus, the cylinders forming the QSM geometry, and other similar small geometric primitives [16], can be used directly as blocks. However, it would also be possible to divide the cylinders and form even smaller blocks. In the case of voxel-based structure models a pre-processing step is required to form blocks that are a collection of voxels. Similarly, in continuous surface models the branch surfaces should be divided into smaller sections that can be used as blocks.

As the leaf insertion algorithm is designed to be as general as possible, i.e. any user-defined distribution can be used, validation can take various forms. We carried out initial validation using leaf area and count measurements from three English oaks together with their QSMs reconstructed from TLS data. Both the TLS and leaf measurements are presented in §2.1. The structure reconstruction process to create the required cylindrical QSMs is briefly described in §2.2.

The leaf insertion algorithm is presented in §2.3 together with the related distributions that control leaf position, size and orientation. Although this paper focuses on sampling the described distributions to produce individual leaves with a known geometry, it is not always necessary, as discussed in §2.4. Section 2.4 shows how the distributions define a leaf density distribution around the structure model blocks, and how that overall distribution can be used for computations without generating the geometry of individual leaves. Although we focus on broadleaves, the procedure can also

Table 1. Leaf area and count measurements.

tree/layer	leaf area (m^2)	leaf count
small oak	153	47 644
0.0 – 11.5 m	18	5432
11.5 – 19.6 m	135	42 212
medium oak	215	52 416
0.0 – 9.0 m	46	12 753
9.0 – 19.9 m	169	39 663
large oak	339	114 224
0.0 – 8.0 m	61	16 056
8.0 – 13.0 m	23	9399
13.0 – 18.4 m	49	19 597
18.4 – 22.4 m	206	69 172

be used for generating needles. Approaches for working with needles are presented in §2.5.

A Matlab implementation of the algorithm, including descriptions of the related classes and the main function, is introduced in §2.6. The Matlab implementation was used to compute several leaf distributions for the oak trees. The results are presented in §3. A discussion is included in §4 and conclusions are made in §5.

2. Material and methods

2.1. Laser scanning and leaf measurements

Our analysis was based on raw point-clouds recorded at Alice Holt Forest, UK (51.1533 N, 0.8512 W), by a single-return phase-shift Leica HDS-6100 TLS (Leica Geosystems Ltd, Heerbrugg) on three 80-year-old oak trees (*Quercus robur* L.). Scans were performed in March 2014, during winter time, under dry and low wind speed (less than 1 $\mathrm{m\,s^{-1}}$) conditions. Trees were recorded from six scan positions around each tree (azimuth angle of 0° S, 60°, 120°, 180°, 240° and 300°) at a distance of 5 m from the base of the tree and with a TLS sampling resolution level of 0.018° at each scan position. Six reflective targets were set out around each tree to merge the multiple scans. Three-dimensional reconstructions of the trees were then computer-generated using the method described in [5].

The trees were harvested in June 2014. The foliage sampling method consisted of a manual stripping-off of each leaf from the branches and storage in bags labelled with the height stratum to which they belonged (table 1). A second component of the method involved the collection of a set of 100 leaves at random from each stratum on each tree. Each stratum bag was then fresh-weighed (Avery Berkel HL206, UK) and oven dried at 75°C to obtain their dry masses. From the subsets, individual leaf area was measured in the laboratory with a laser area metre (CID-203, Camas, WA, USA) and weighed (Mettler Toledo AG204, Switzerland) before and after oven drying at 75°C. Specific leaf area (SLA) was derived for each of the subsets and used to estimate the total leaf area and the number of leaves for each stratum (e.g. [12,17]). Additionally, the average area of the leaves was recorded from the smallest to the largest tree as 33.71, 40.33 and 29.66 cm^2, respectively.

2.2. Quantitative structure models

The three oak trees were reconstructed as cylindrical QSMs in Matlab with the procedure detailed in [18]. The properties of the resulting models are listed in table 2. Furthermore, the branch count distribution per branch order is visualized in figure 2. The count of the branches is important as leaves are placed near the tips of the branches.

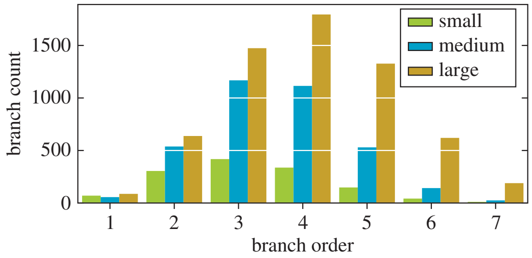


Figure 2. Branch order–count distribution. The stem and branch orders 8 and 9 have been excluded due to their negligible portions. (Online version in colour.)

Table 2. Oak tree properties computed from reconstructed QSMs.

	oak tree		
property	small	medium	large
branch count	1334	3579	6161
cylinder count	8429	23 539	35 428
DBH (mm)	298	432	848
height (m)	19.1	19.6	21.8
order max.	9	8	9
total length (m)	592	1552	2516
volume (l)	707	1169	2098

The small and medium oaks were similar in height, but the latter had about 2.6 times more branches when measured in total count and in length. The large oak had the most branches for all branch orders, and almost twice the volume of the medium oak.

2.3. Leaf generation algorithm

This section describes an algorithm to populate QSMs with leaves. The main inputs of the algorithm are distributions that control the position, orientation and size of the leaves. These distributions are sampled to retrieve the parameters of individual leaves. The approach can be described as simplified or naive, for three reasons: (i) position, orientation and size are sampled independently, which is to say that, for example, the size of a leaf may not affect its orientation; (ii) simple controls for phyllotaxy and clumping effects are yet to be implemented (although there is some control when generating the petioles); and (iii) the only effect leaves have on one another is that they are prevented from intersecting. We call this procedure the foliage and needles naive insertion algorithm, or the FaNNI algorithm in short.

2.3.1. Overview of the procedure

The inputs of the algorithm are a collection of QSM blocks, leaf basis geometry, target leaf area to be distributed, and petiole and leaf parameter distributions. Details of the roles of the leaf basis geometry and the distributions are presented in §§2.3.2 and 2.3.3, respectively. The process can be viewed as two separate stages: (I) generating candidate leaves and (II) accepting candidates while preventing intersections. An overview of the process is provided in figure 3.

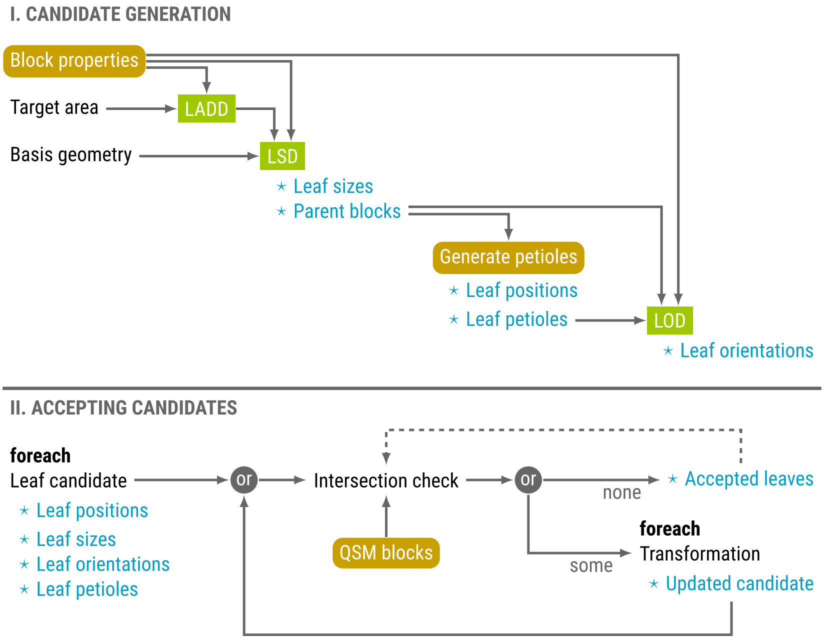


Figure 3. Process overview of the leaf generation process. Leaf distributions are drawn in green, and functions and properties related to the QSM in orange. The main outputs are written in blue. The two stages are presented on top of one another. (Online version in colour.)

The first stage begins by distributing the available leaf area onto the blocks. The leaf area density distribution (LADD) determines the relative probability for a block with given parameters to have leaf area. After sampling the distribution with the block properties, each block has a target leaf area, or a leaf area budget, that will be divided into individual leaves by sampling the LSD.

For the leaf size determination the blocks are processed in random order. To match the target leaf area as closely as possible the cumulative area difference with respect to the target is updated after each leaf. While there is room in the current block, or the cumulative area budget, a new leaf is added to that block. The algorithm assumes that all the generated leaves have the same geometry, and thus we can sample a leaf length value which can be converted to area. After this step, the number of generated leaves and the block parent of each leaf are known.

Next, the locations of the leaves are determined by physically attaching them to their branches by the petioles. Because TLS measurements usually cannot capture petioles as they are too small to be detected reliably, all the petioles are generated: the petiole's starting point, orientation and length are determined by sampling appropriate parameter distributions given by the user. The end point of a petiole also determines the origin of the respective leaf. Although the exact petiole geometry is computed, they are considered insignificant compared with the blocks and the leaves, and thus they are excluded later from the intersection detection process.

The final property to sample is the leaf orientation. The LOD is used to determine the direction and the surface normal of each leaf. Once this is done, all leaves have a fixed position, orientation and scale, and their geometry can be computed by transforming the leaf basis geometry accordingly.

At this point it is possible, and even likely with a high leaf count, that some of the leaf candidates intersect one another, or the blocks, as they were generated independently. However, the goal is to produce a model without leaf intersections, and thus in the second stage the leaves are checked one by one for intersections before adding them to the list of accepted leaves.

If a leaf candidate intersects a block or an accepted leaf, it is possible to try to change the position, orientation and scale of the leaf and check whether the intersection was avoided. If it was, the leaf candidate is accepted; if not, the process can be repeated any number of times with a different transformation applied to the parameters. If, despite all the transformations, intersections cannot be avoided the candidate is discarded. An example of how intersection prevention can be implemented is described in §2.3.4. The leaf generation process stops when all the leaves have been processed, unless some other stopping condition has been given, such as a target leaf area of accepted leaves.

2.3.2. Leaf model

The leaf model defines the basis geometry of an individual leaf. This geometry is the same for all the sampled leaves, but it is scaled, rotated and translated to receive the final leaf geometry, during the generation process. Thus all the generated leaves have the same shape but the size and orientation can vary. In the simplest case, the basis geometry can be a single triangle, allowing fast leaf cover generation due to simple intersection detections. For examples of basis geometries consisting of triangles, see §2.6. On the other hand, there is no upper limit for the complexity of the basis geometry, other than computational time requirements to ensure non-intersecting leaves. Thus, it is possible to represent more complicated shapes, e.g. a leaf with three-dimensional curvature, or a compound leaf with several leaflets, that do not have to lie on the same plane. However, to simplify the generation process, it is possible to use a simplified basis geometry while generating the leaves, which is then replaced with something more complex, as long as the change does not introduce additional intersections.

The origin of the leaf basis coordinate system is assumed to be the point where the petiole connects to the leaf. Leaf *direction* is the direction from the origin towards the tip of the leaf, and perpendicular to this lies the leaf *normal* that defines the direction to which (most of) the leaf area is facing. The length of the basis geometry, i.e. leaf length, is fixed at unity. Other dimensions are given with respect to that. During leaf parameter sampling only the leaf length is sampled as it determines the leaf area when the basis geometry is fixed. Note that it is not required to compute the exact geometry of the leaf candidates before the intersection prevention stage.

2.3.3. Leaf and petiole parameter distributions

Leaf and petiole properties are controlled by multiple user-definable distributions which are sampled when leaves are generated. The properties fix the number of leaves, their position, size and orientation. In theory, these distributions are multidimensional as they may depend on any number of block properties, such as height from the ground, radius and orientation. They can also be formed as a weighted product or sum of one-dimensional marginal distributions. The purpose of each distribution is described below in the order they are sampled in the implementation.

2.3.3.1. Leaf area density distribution

Total leaf area is one of the inputs of the algorithm, and leaf area density distribution defines how that area should be distributed to the blocks. Thus, the leaf area density distribution can allocate more leaf area towards the top of the tree and towards the tips of the branches. One could also prevent leaf area from being attached directly to stem blocks by using branch order information. Furthermore, the distribution produces a relative mapping of area on the blocks, allowing the distribution to assign any given total area of leaves to the structure model.

2.3.3.2. Leaf size distribution

After a leaf area target has been assigned to each block, the LSD is used to sample leaf count and size, so that the target area is matched as closely as possible. This distribution determines the number of leaves to be generated N_{init}. However, as no intersections between leaves or between blocks and leaves are tolerated, the final number of leaves may be smaller than initially generated if intersection cannot be avoided with transformations, i.e. $N_{\mathrm{final}} \leq N_{\mathrm{init}}$ holds.

2.3.3.3. Petiole generation

After size distribution sampling, the number of leaves is known and it becomes possible to sample the petioles that connect the leaves to their block parents. Similarly to leaves, petiole parameters include the starting point, orientation and length of the petiole, which effectively also determine the starting points, or origins, of the leaves. It would be possible to model the petioles as three-dimensional objects, like small cylinders, but the implementation considers them only as line segments, and they are excluded from the intersection prevention step.

2.3.3.4. Leaf orientation distribution

The final distribution controls the orientation of the leaves. This distribution controls the directions and normals of the leaves, and can be used to describe, for example, which parts of the tree are *erectophile* and which are *planophile*.

2.3.4. Intersection prevention

Sampling the presented leaf and petiole parameter distributions results in a list of N_{init} candidate leaves. But because each sample is independent of the rest, the leaves may intersect with other leaves in the list, or blocks of the QSM. To avoid intersections, leaves are only accepted to the final collection of leaves if they do not intersect with other geometry.

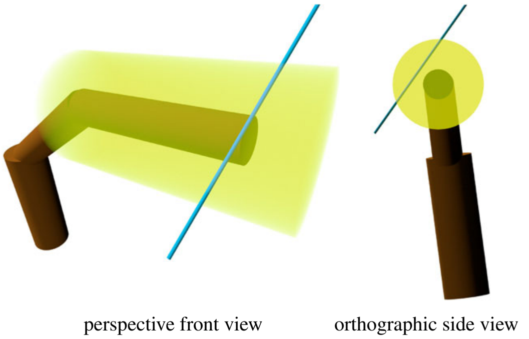


Figure 4. Two views of an example ray (blue) travelling through the leaf density cylinder (yellow) that is supported by one of the branch cylinders (brown).

The accepted leaves list is initialized as empty. One by one, the initial leaves are checked, so that they do not intersect with any of the blocks or the accepted leaves. To avoid a low acceptance rate, an intersecting leaf is not discarded instantly. Instead, a number of preselected user-defined transformations are applied to the leaf candidate, and intersection checking is repeated. A transformation may consist of any combination of scaling, rotation and translation, but they are applied in that order. Only if none of the preselected transformations prevent all the intersections, the candidate is discarded.

2.4. Leaf density model

Section 2.3 described an algorithm to generate exact leaf geometry by sampling certain distributions that depended on individual block parameters. However, in some cases it is not necessary to compute the exact geometry, but rather to view the leaves as an abstract density around the branches [19]. Such an approach saves computational resources as there is no need to compute and store a lot of geometry. This is especially relevant for computations with needles as their number often far exceeds the number of broadleaves for similar sized trees. This abstract approach without exact leaf realizations can be suitable for many applications, e.g. ray tracing operations in radiative transfer and gap fraction computations. However, exact geometry may be better suited for some applications, e.g. requiring realistic visualization, and it is also a more straight-forward way to study effects on a single broadleaf of needle scale.

The distributions defined earlier depended on block properties, which essentially means that each block defines a density, size and angle distribution around itself. In the case of a cylindrical QSM, this can be viewed as a *leaf density cylinder* around the block (figure 4). The radius (and length) of the leaf cylinder is defined by petiole length and LSDs. Let us next briefly justify the leaf cylinders as potentially useful and consider ray tracing with leaf cylinders as an example. One possible approach for ray tracing applications would be to determine an absorption rate for the leaf cylinder, which can depend on the distance from the cylinder axis, and where the rate can be stochastic (cf. the turbid medium analogy [4]). Branch cylinders can be viewed as infinitely dense, and thus hits occur at their surface. When enough of the energy of a simulated beam is absorbed, a hit occurs inside a leaf cylinder. If the application requires it, an incidence angle can be sampled from the orientation distribution stored in the respective block.

2.5. Inserting needles

Although this paper focuses on demonstrating broadleaf insertion, it is possible to use the algorithm with needles in different ways. The most obvious method is to use a tiny cylinder

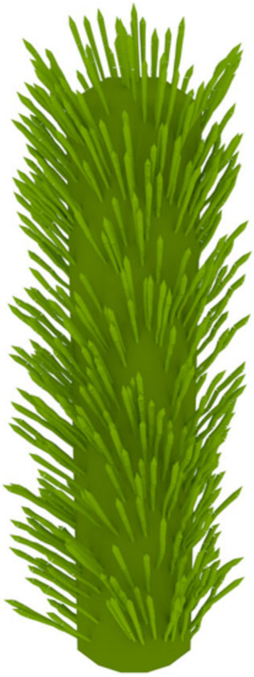


Figure 5. An example of a needle bud three-dimensional model without a strict phyllotaxy. (Online version in colour.)

to represent a single needle and use that as a basis geometry. However, the computational requirements of the insertion would be enormous (but not impossible [20]), as they would be for any further application using the resulting model.

A less resource-consuming approach would be a modification of the leaf density cylinder approach described in §2.4. Rather than inserting needles at all, they could be viewed as a density distribution around the blocks (cf. [19]). Note that the distribution does not have to be uniform, and thus it can be used to account for needle phyllotaxy. Additional buds could also be introduced as density cylinders if the QSM does not contain the level of detail in terms of branching structure required by the user. Even though exact needle geometry is not generated, it is important to incorporate the needle phyllotaxy in any ray tracing operations inside needle density cylinders, as it is key in simulations including needles [21].

A third option would be to use a needle bud as the basis geometry. An example of a needle bud suitable for visualization applications can be seen in figure 5. Even though the model is complex, it can be simplified to a cylinder during the intersection checking stage. The complex model can still be used for visualizations, or in further computations when required.

2.6. A Matlab implementation

The leaf insertion algorithm was implemented in Matlab [22]. The supporting classes and the main function of the implementation are presented below. Currently the implementation works with leaves, where the basis geometry is a collection of triangles, and cylindrical QSMs, but the structure of the implementation is modular, so that it is easy to extend to other types of leaves and blocks as necessary.

2.6.1. Classes

The following classes were written to make the implementation as modular as possible. Especially, the `LeafModel` and `QSMB` abstract classes were designed to define interfaces for easy extendibility when using other structures than cylindrical QSMs, or triangle-based leaf models.

2.6.1.1. LeafModel

The objects of this class have two main purposes in terms of the data they hold. First, they contain the leaf basis geometry, which is transformed to determine the geometry of the generated leaves. Second, they hold the parameters of the accepted leaves, i.e. leaf origin, scale, direction and normal. In terms of functionality the class is responsible for defining an intersection detection method for two leaves. There is also a method for converting the geometry of a leaf into a collection of triangles. The `triangles` method is required mainly when detecting intersections between a leaf and a block.[1] There is also a method for adding a new, accepted leaf to the model.

`LeafModel` is an abstract class, used only for defining the required interface for subclasses rather than actually creating instances. This allows the class to be extended by creating subclasses, such as the implemented `LeafModelTriangle` class for leaf models, where the leaf basis geometry consists of vertices and triangular faces. This class already allows numerous leaf shapes, as seen in figure 6, but the user can extend the possibilities by implementing a subclass of `LeafModel`, e.g. for leaf geometry defined with Bézier curves, or other vertex–face-based geometries but with more optimized intersection detection than checking each triangle separately.

2.6.1.2. QSMB

The class name is an acronym for quantitative structure model blocks (QSMBs), and it essentially acts as a container for QSMB information. The class is abstract and used to define an interface for its subclasses. The interface includes a method for reading block properties, such as position, orientation and branch order, and to detect intersection between blocks and triangles. Furthermore, a `QSMB` object is responsible for generating the petioles of the leaves using the block geometry. Finally, there is a method for converting the blocks of a QSM into a `CubeVoxelization` object, which is used to optimize intersection detection.

As an example subclass, the `QSMBCylindrical` was created to contain cylindrical QSM data. In this class, the block data consist of cylinder parameters for the geometry, and branching topology, such as branch order information. The user can extend the implementation to work on other types of structure models, by providing the appropriate subclass definition.

The `QSMBCylindrical` class also defines default uniform distributions for the petiole parameters. In this initial implementation, the petiole parameters are the following, with the lower and upper limits in parentheses: relative position along the cylinder axis $(0, 1)$; relative position in the radial direction when connected to the end circle of the last cylinder in a branch $(0, 1)$; rotation around the cylinder axis $(-\pi, \pi)$; petiole elevation $(-\pi/2, \pi/2)$; petiole azimuth $(-\pi/2, \pi/2)$; and petiole length (2 cm, 5 cm).

2.6.1.3. CubeVoxelization

An object of this class is a voxelization of a fixed three-dimensional space into cubical voxels with a fixed edge length. A `CubeVoxelization` object has a minimum and a maximum point and the space between them is divided into a finite number of cells. Object references can be stored in the cells to indicate that the objects occupy at least a part of that voxel. In the main function of the leaf insertion implementation, voxelizations are used to store and find candidate leaves and blocks, to perform more accurate intersection detection. Furthermore, the edge length of the voxelizations is set as the maximum leaf size produced by sampling the LSD function.

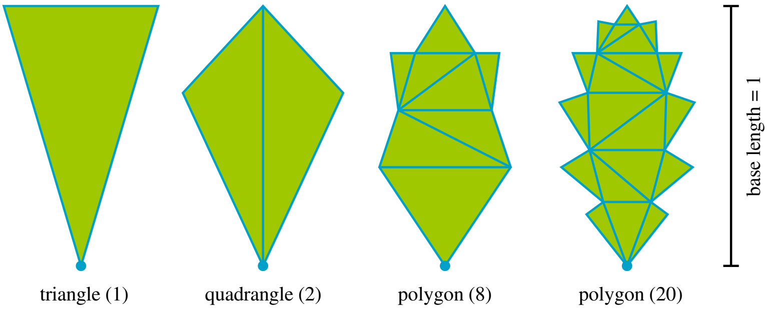


Figure 6. Triangular basis leaf geometries. The number of triangles is given in parentheses. The origin of the leaf is marked with a circle, and the length of a basis geometry always equals 1. (Online version in colour.)

2.6.2. Main function

qsm_fanni is the main function that receives the QSM as a QSMB object, an initialized LeafModel object that contains the leaf basis geometry, and total leaf area to be distributed. The leaf area parameter can have two components; one for the initial leaf area A_{init} to be generated, and one for the target leaf area $A_{\mathrm{target}} \leq A_{\mathrm{init}}$. This can be used to increase the probability that the target area is reached, even if some of the generated leaves are discarded due to unavoidable intersections.

There are also numerous optional inputs for the user to customize, such as the distribution functions and transformations during the intersection prevention step. However, default options are available for all the remaining parameters.

The main output of the function is a LeafModel object derived from the corresponding input, but it now contains the accepted leaves, petiole start points and a vector of parent block indices of each accepted leaf.

2.6.3. Default leaf parameter distributions

The implementation contains default distribution functions for leaf parameter properties, and they are described below. At the moment these defaults are not designed to be biologically accurate, but rather just to provide an example of distributions. However, there are plans to improve the realism and usability of the default options in future versions, by offering the user a choice between common options, such as a spherical distribution for the leaf orientation.

2.6.3.1. Leaf area density distribution

By default the available leaf area is distributed equally to all the last cylinders in the branches of the QSM. All other cylinders remain leafless.

2.6.3.2. Leaf orientation distribution

The default LOD is such that most of the leaf area faces upwards, but there is some random variation. The LOD computes an initial leaf normal estimate as a cross product of the petiole direction and a side direction on a horizontal plane. If the initial direction differs by less than 20° from a reference direction (straight up in this case), then the final normal direction is the reference direction. Otherwise, the final normal is the initial direction rotated towards the reference direction by 20°.

2.6.3.3. Leaf size distribution

The default LSD samples a leaf length value from a uniform distribution with given limits. That value is then scaled with a value based on the relative height of the parent block to ensure that leaves are a little bit larger at the top of the tree.

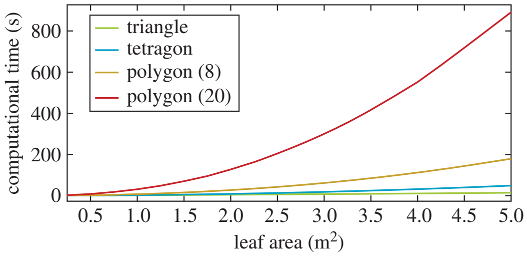


Figure 7. Computational time as a function of total generated leaf area for a single test cylindrical block. The values are averages over the 10 repeats.

3. Results

3.1. Leaf geometry complexity test

The LeafModelTriangle class enables the use of leaf basis geometries with an arbitrary number of triangles. However, the detection of intersections between leaves requires that all those triangles are checked, which has an enormous effect on computational time. To study the effect of the number of triangles on the basis geometry, a single cylindrical block (length 1 m, radius 0.25 m) was fitted with an increasing total area of leaves. The area varied from 0.25 to 5 m^2 for the four basis geometries in figure 6. The process was repeated 10 times for each leaf area–basis geometry pair. The average computational time results are shown in figure 7.

When using a single triangle, generating non-overlapping leaves was very fast even with the maximum leaf area, 5 m^2, taking only 11 s on average. With the two-triangle quadrangle, the times increased 1.8-fold to 4.3-fold in comparison with the single triangle when moving from the lowest to the highest leaf area. For the polygon with eight triangles, the required time was 8.1-fold already at 1 m^2 and 16-fold at the maximum. The respective multipliers for the 20-triangle polygon were 35.9 and a 79.7, which translate to 31 and 891 s, respectively.

3.2. Leaf area density distribution definitions

To demonstrate the leaf insertion algorithm, we defined the two following parametrized leaf area density distributions. While we tested other distributions and parametrizations, these two were chosen because of the low parameter count and overall simplicity.

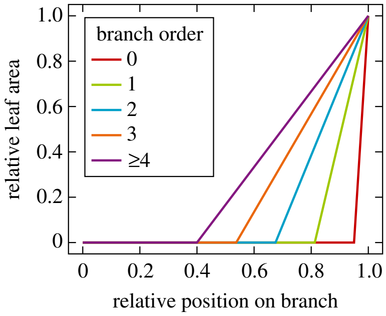


Figure 8. Piecewise linear polynomials defining the branch order-dependent LADD 2 scaling factor $y_4 = 0.4$.

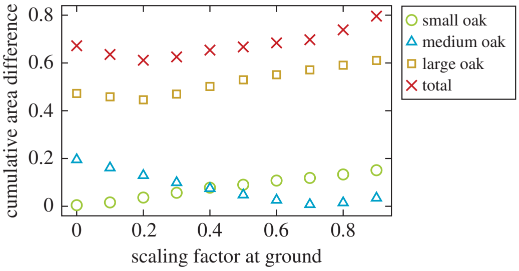


Figure 9. Cumulative area difference curves for the LADD 1 distribution as a function of the height scaling parameter. (Online version in colour.)

LADD 1 initialized the last 5% of each branch to have an equal portion of leaves, then scaling these proportions with a factor dependent on the relative height of the respective cylinder. The factor had a value of the parameter y_0 at ground level and 1 at the top of the tree. Values in between were interpolated linearly.

LADD 2 had an additional parameter to define a cut-off point along a branch. The branch did not have any leaves before this point, which was dependent on the branch order. For the stem the cut-off was at 95%. For branch orders 4 and above, the cut-off was at y_4, and for lower branch orders the cut-off was interpolated linearly. For cylinders after the cut-off point, the probability of leaves was interpolated linearly between 0 at the cut-off and 1 at the tip of the branch. Furthermore, the probabilities were scaled with a factor depending on the relative cylinder height as with LADD 1. The scaling factor y_4 is visualized in figure 8 for a parameter value of 0.4.

To find the optimal values for the parameters, we performed a simple grid search by varying the values of y_0 and y_4 in the closed intervals $(0, 1)$ and $(0, 0.9)$, respectively. For LADD 1, which only depends on the y_0 parameter, the results are shown in figure 9; for LADD 2, the optimal parameter values are listed in table 3. Optimization was done on the cumulative area difference that was computed as the sum of unsigned leaf area differences in the vertical layers of the trees. The error was normalized with the measured total leaf area of the tree. The total error was computed as a sum over all the trees.

For LADD 1 the total optimal value was $y_0 = 0.2$, which was close to those of the small and large oak trees. However, the optimal value of the medium oak tree was different at 0.7. For LADD 2 the total optimum values were $y_0 = 0.2$ and $y_4 = 0.5$, but there were differences in the optimal parameter values between the individual trees.

Table 3. Optimal parameter values for LADD 2 distribution. Parameter y_0 controls the vertical distribution and parameter y_4 the distribution along the branch length.

tree	y_0	y_4
small oak	0.1	0.7
medium oak	0.6	0.5
large oak	0.2	0.9
total	0.2	0.5

Figure 10 visualizes the LADD 2 distribution with the optimal parameter values on the small and medium oak trees. Grey parts have no leaves, green parts have some, and red parts have a lot of leaves. Furthermore, figure 11 shows similar LADD heat maps and corresponding generated leaves. Note that in figure 11 LADD 1 is the same as LADD 2 with parameter value $y_4 = 0.95$. Going from top to bottom the regions of high probability of leaves spread from the very tip towards the base of the branch. In the top two rows, the leaves are very concentrated at the tips, whereas in the latter two the leaves are more evenly spread along the high-order branches.

3.3. Leaf insertion test for oak trees

Each of the three oak trees was inserted with their measured leaf area (highlighted in table 1). The two LADDs described above with the optimal parameters were used, and all tree–LADD pairs were repeated 10 times. As we lacked reference data for the leaf orientation and LSDs, defaults from the Matlab implementation were used. To match the measured leaf sizes for each tree, the limits for the default uniform leaf length distribution were derived from the average leaf area measurements. The mean leaf length l_i for tree i was computed as follows:

$$l_i = \frac{\sqrt{A_i}}{r/2}, \tag{3.1}$$

where A_i is the average leaf area for tree i, $r \approx 0.6$ is the ratio between the width and length of the leaf basis geometry, which in this case was the quadrangle from figure 6 to keep the triangle count low. The leaf length limits were computed for each tree as $l \pm 1$ cm.

The computations were done on a quad-core computer (Intel Core i7-6700 K 4 GHz, 32 Gb RAM). The computational mean times and standard deviations over the 10 repeats are listed in table 4. The average computational time per QSM block was between 20 and 40 ms for all the trees. Most of the computational time (95.3%) was spent on detecting intersections, which further supports using the simplest possible leaf basis geometry. The table also lists the average number of required block and leaf neighbour computations, the average number of performed transformations to avoid intersections, and the discarded leaf candidate percentage. The small oak tree had twice the leaf area per branch in comparison with the other two trees, which explains why there were twice as many neighbouring leaf computations and discarded leaves. The results suggest that it would be sufficient to sample 5–10% more leaves than the target leaf area to account for discarded leaves. The results show that the vast

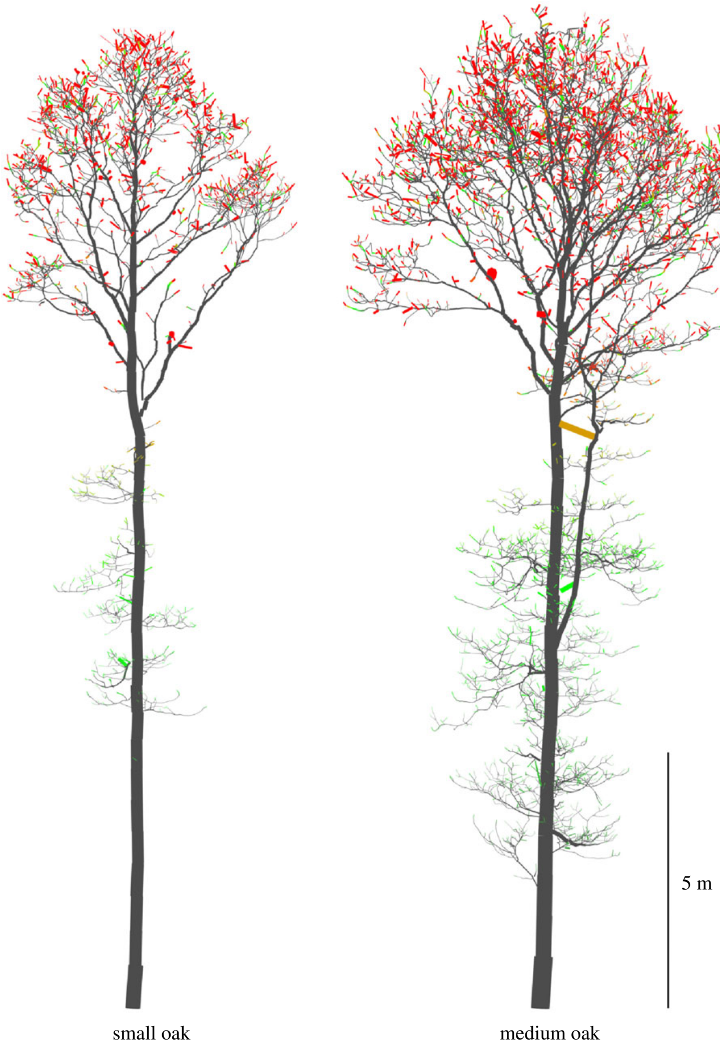


Figure 10. Example leaf area density distribution (LADD 2) for the small and medium oak trees as heat maps. As branch tips are small in size all cylinder radii have been scaled up to four times larger according their LADD value for a better visualization.

majority of leaf candidates are accepted without any transformation as the average number of tried configurations was between 1.0 and 1.5 for all the trees.

Figure 12 shows a top view of all the oak trees with leaves generated with both LADDs, and figure 13 shows a side view of the LADD 1-generated leaf covers for the medium and large oaks. The differences between the leaf covers generated with LADD 1 and LADD 2 are subtle, but notable. As the higher order branches have a lower cut-off point along the relative position on the branch, leaf cover is more even, making the gap fraction smaller on LADD 2 covers.

To compare the generated leaf distributions with the measured data, the leaves were placed in the same vertical bins listed in §2.1 according to their centre. The signed difference between generated and measured leaf count and area are listed in table 5. Negative values mean that the tree or layer should have had more leaves or leaf area; positive values are the opposite. Both LADDs were able to match the measured leaf area at the tree level because that was the stopping condition. The tree-level leaf counts are only between 500 and 3500 below the target values. Relative to the total leaf count the differences were 7.5%, 0.9% and 2.0% for the small, medium and large oaks, respectively.

The layer-level differences were much higher, which suggests that the vertical distribution generated by the proposed LADDs did not match the measurements. With LADD 2 the top layer of the large oak was missing over 90 m^2 of leaf area while the layer below that had an excess of about 60 m^2. Results for the small oak were similar, which suggests lowering the y_0 parameter. However, the opposite was true for the medium oak, which had about 6 m^2 of extra leaf area in the upper layer.

4. Discussion

The above results presented two relatively simple LADD functions that used branch order, relative height and relative position along a branch to determine the portion of leaf area to be assigned to a block. However, the implementation

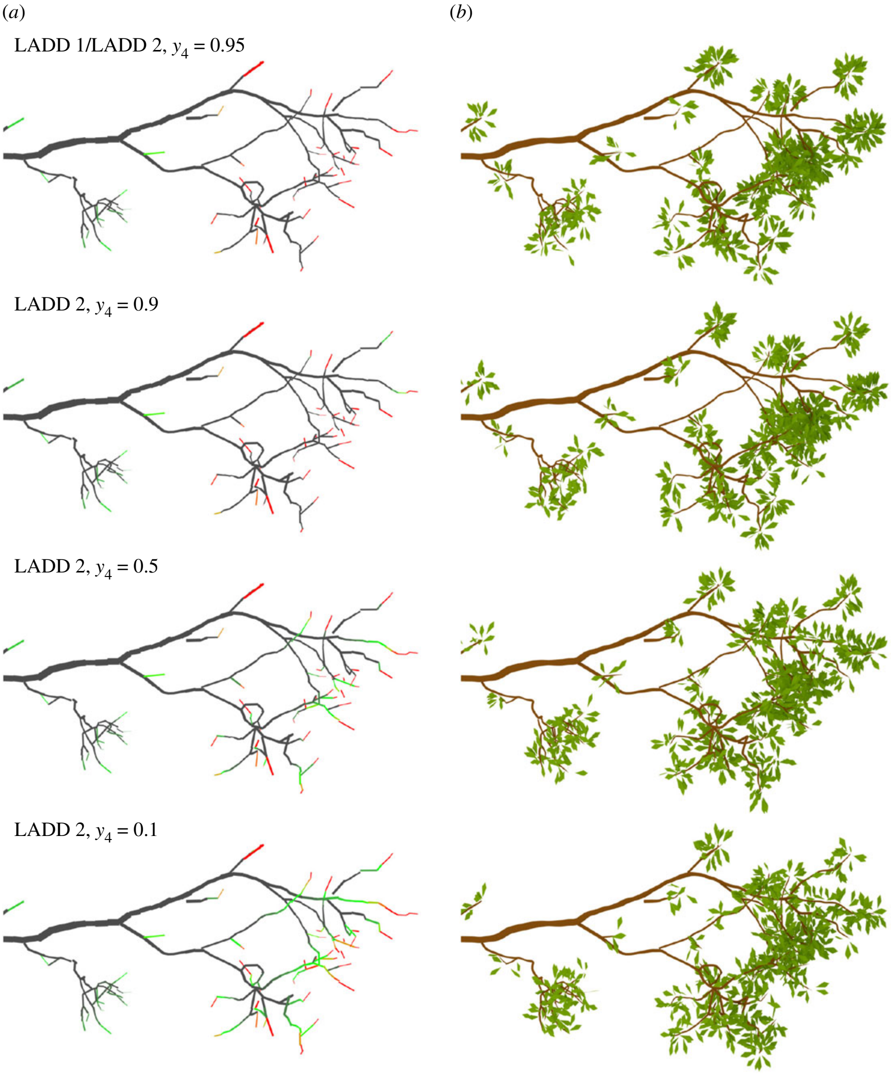


Figure 11. LADD examples on a single branch from the small oak tree. The distributions control how leaf area is distributed on the supporting branching structure. The parameter y_4 controls the cut-off point along the branch length, starting from the branch base, before which there can be no leaves. (*a*) Distribution as a heat map; (*b*) sampled leaves based on the corresponding heat map.

Table 4. Oak tree average leaf generation results. The properties are computational mean time, time standard deviation, average block and leaf neighbour counts, and average number of transformation configurations tried before accepting or discarding a leaf.

tree/LADD	time	time std	block neigh.	leaf neigh.	transforms	discard (%)
LADD 1						
small oak	6 min 12 s	7 s	13.1	32.8	1.4	7.3
medium oak	7 min 55 s	9 s	15.7	16.3	1.0	3.4
large oak	17 min 48 s	30 s	11.8	16.2	0.9	3.5
LADD 2						
small oak	6 min 32 s	4 s	13.6	33.9	1.4	7.8
medium oak	8 min 07 s	5 s	16.1	16.2	1.0	3.6
large oak	18 min 19 s	8 s	12.4	16.5	1.0	3.6

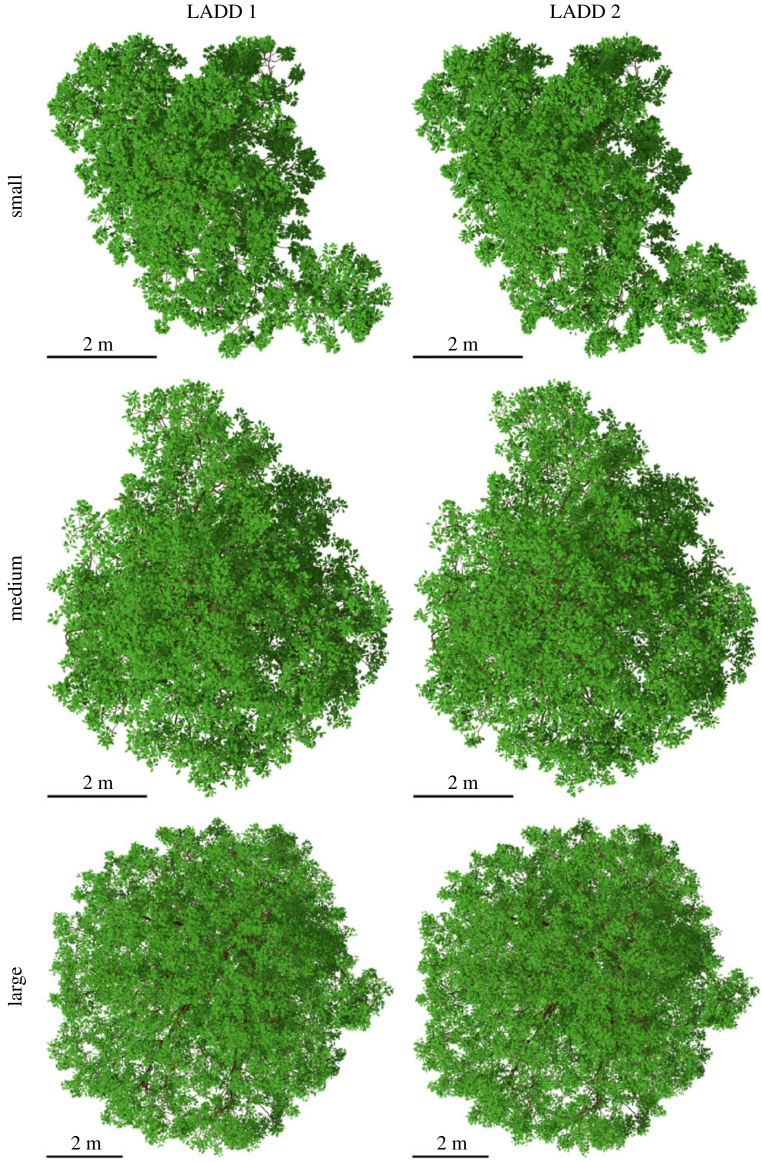


Figure 12. Top view of the three oaks with leaves generated with the two LADDs. (Online version in colour.)

allows for the user to write more complex LADD functions that make use of additional information, such as absolute height (whether the block is above the surrounding canopies) and absolute orientation (north or south side of the stem). Owing to limited reference data only the LADD was optimized. However, if detailed leaf angle or leaf size measurements are available, it is possible to optimize the respective distribution in a similar manner.

The LADD parameter optimization results and the conflicting layer difference results show that the presented LADDs are not able to capture the differences in the leaf area distributions of the three oak trees. Further studies should be carried out to assess whether the underlying leaf distributions differ between these three trees, or whether it is simply a matter of choosing a better LADD. It should also be noted that the manual leaf measurements were limited with only eight data points in total for the three trees, and, as such, more detailed and comprehensive measurements would be beneficial. Some of the leaf area difference can also be explained by uncertainties in estimating leaf area and count for the vertical layers, and by missing branches in the upper canopy in the QSMs.

The parameters of the two LADDs were optimized by using a grid search where exact leaf geometry was generated at each grid position. This made the optimization computationally intensive as 95% of the computational time was spent on intersection prevention, which forced a low parameter count. However, in retrospect it was unnecessary to generate leaf geometry, because as the results showed the discard rate was very low, which means that the LADD of the output was very close to the input. Thus, optimization according to, for example, vertical layers can

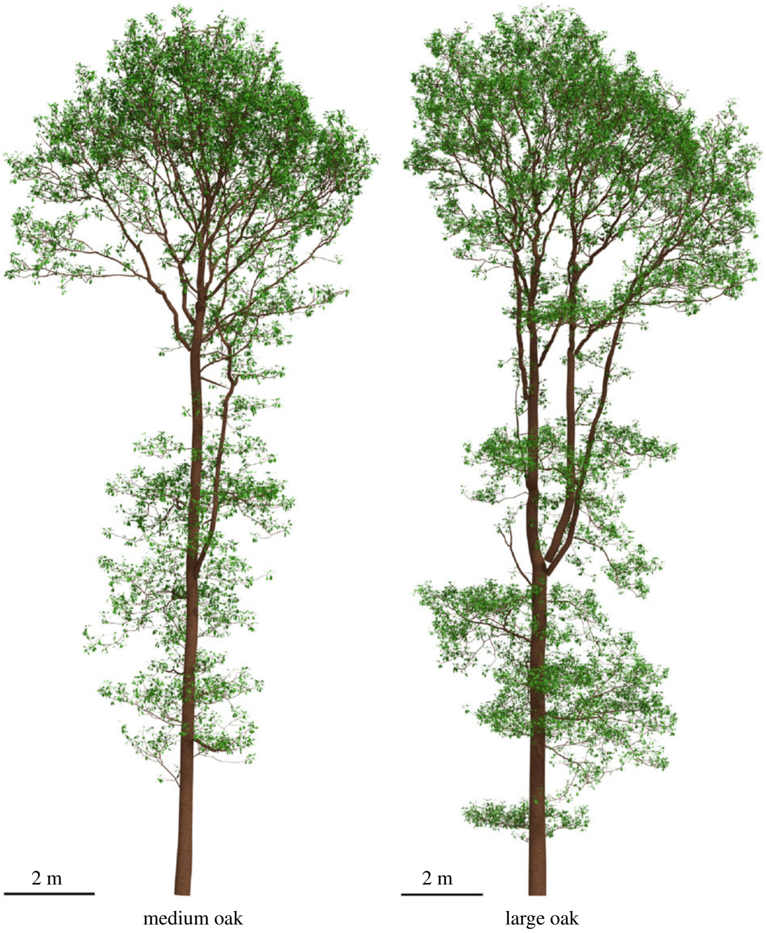


Figure 13. Side view of the medium and large oak with leaves generated with LADD 1. (Online version in colour.)

Table 5. Difference between oak leaf count and leaf area in total and in vertical layers.

	LADD 1		LADD 2	
tree/layer	Δ count	Δ area (m^2)	Δ count	Δ area (m^2)
small oak	−3561	+0.0	−3581	+0.0
0.0–11.5 m	+1707	+5.7	+1002	+3.3
11.5–19.6 m	−5268	−5.7	−4583	−3.3
medium oak	−473	+0.0	−432	+0.0
0.0–9.0 m	−3339	−8.1	−2811	−6.0
9.0–19.9 m	+2866	+8.1	+2379	+6.0
large oak	−2275	−0.1	−2157	+0.0
0.0–8.0 m	+9507	+12.9	+10 748	+16.6
8.0–13.0 m	+2758	+13.1	+3254	+14.5
13.0–18.4 m	+15 634	+58.7	+15 883	+59.4
18.4–22.4 m	−30 174	−84.8	−32 040	−90.5

be simplified to only include distributing the available leaf area onto the structure model and exclude both leaf size and orientation sampling and especially the computation of exact geometry.

Future research should also include testing the importance of the intersection prevention for various applications, i.e. whether possibly intersecting and non-intersecting leaves differ significantly in terms of required resources and produced level of detail. This way we would know whether it is sensible to perform the intersection prevention step, e.g. for simulations studying light use efficiency.

In this paper, the proposed method was only used to generate leaf covers according to user-given distributions. However, it would also be interesting to see whether this algorithm could be used to invert or approximate the real-leaf distributions of a given tree, with simple non-destructive and non-direct measurements. For example, it would be possible to test whether gap-fraction measurements and suitable parametrizations of the leaf distributions can be used to optimize the distribution parameters, to derive a mathematical or even a biological explanation for the real leaf distribution. With this method, it is possible to make such simulations and study this inverse problem. It should be noted that such inversion does not reconstruct exact leaf geometry but rather gives an approximation of their distributions. Such an approach could produce new understanding of what affects the distribution of leaves for a specific tree. Furthermore, it would allow the generation of leaf covers that follow the reconstructed distribution for the same tree or some other tree.

Currently the algorithm views each leaf independent from the others (apart from intersection prevention), which is one of the reasons for calling the algorithm naive. However, in most tree species leaves follow a certain phyllotaxy or the leaves are clumped together, e.g. their petioles originate near one another, or even from the very same spot [23]. We are planning to implement simple phyllotaxy controls in future versions of the FaNNI implementation. The level of clumping could be defined as a separate distribution that would be used to sample the size of a clump and variation in petiole and leaf parameters for the leaves within the clump.

In nature, leaves are often connected to branches that are small in diameter. Because of the limitations of the TLS technology, such branches are often poorly sampled in the resulting point clouds. Therefore, they can be excluded from the reconstructed QSM also, which means that, when leaves are inserted, they are connected to branches that are too large. To counter this shortcoming, it is possible to perform a pre-processing step that inserts small branches into the structure model, which will be given a high probability of leaves when defining the LADD function.

Although the implementation enables the use of leaf basis geometries consisting of any number of triangles, the results show that additional complexity multiplies the expected computational time by large factors. However, if detailed leaf geometry is required for later computations, it is possible to use a simplified stand-in basis geometry that encapsulates the complex shape to prevent overlapping during generation and replace the geometry afterwards. Such a procedure could even be built in to an extension of the `LeafModel` class.

5. Conclusion

We have presented an algorithm to generate non-intersecting leaves to a QSM that follow user-defined position, size and orientation distributions. A Matlab implementation of the algorithm was also presented. Currently, the implementation allows the use of any leaf shape consisting of an arbitrary number of triangles.

In order to present leaf property distributions in a compact yet versatile format, we propose a scheme where a QSM is divided into blocks that determine, and can be used to contain, property information for leaves that are to be connected to it. This means that we can assign the available leaf area, leaf size and orientation parameters to the blocks of a QSM even without generating leaves. Then we can do one of the following.

— Visualize the property distributions by colouring the blocks according to their respective property values as seen in the case of leaf area density distributions, e.g. in figures 10 and 11.
— Sample the user-defined distribution with the parameter values and generate exact leaf geometry as was done in §3.
— View the leaves as a probability distribution around the QSM blocks, and rather than computing exact leaf geometry do computations by determining the probability of a hit and the incidence angle when a beam enters the vicinity of a block.

Although any triangle-based geometry is possible for the leaves, a simple test of adding an increasing area of leaves to a single cylindrical block showed that complex leaf shapes can drastically increase the computational time, at least with the current implementation. Thus, the leaf basis geometry should be kept as simple as possible, or optimization is required for intersection detection.

To demonstrate leaf generation, we presented two different LADDs and applied them to three oak trees trying to match field measured leaf count and areas. The measurements were done with two–four vertical bins per tree, and the average leaf area was also recorded for each tree. Simple uniform LSD (with some scaling based on height) and planophile orientation distribution were used, while the main focus was on optimizing the LADDs. The two suggested LADDs were able to match leaf area and count per tree, but the vertical distribution of leaves had major errors despite the optimization. Further research is required to understand the cause of the leaf area differences.

A further goal is to use the leaf-augmented QSM (L-QSM) to incorporate a number of biological principles such as the availability of resources (mass and energy exchanges between vegetation and atmosphere, and phyllotaxy) to construct as many self-consistent tree models as possible. One can include stochastic variations in the same sense as in the creation of four-dimensional QSMs [24], extending that scheme to fully functional trees. This approach would enable a large number of applications to verify and refine assumed biological postulates of theoretical models, and then use the resulting full-scale three- and four-dimensional models for predictions and the modelling of ecological systems at various size and complexity scales, including large-scale statistical (allometric) estimates.

Data accessibility. The Matlab implementation of the FaNNI algorithm is available on GitHub (https://github.com/InverseTampere/qsm-fanni-matlab). The three oak tree QSMs are available from Eric Casella (Eric.Casella@forestry.gsi.gov.uk) upon request.

Authors' contributions. M.Å. developed the FaNNI algorithm, wrote the implementation, carried out the computations and drafted the

manuscript. E.C. acquired the TLS measurements, computed the QSMs and led the destructive leaf sampling experiment. M.D., F.M.D., R.G. and L.A.S. participated in that experiment. P.R., M.K., E.C., M.D., F.M.D., R.G. and L.A.S. helped draft the manuscript. All authors gave final approval for publication.

Competing interests. The authors have no competing interests.

Funding. This study was funded by the Academy of Finland research project Centre of Excellence in Inverse Problems (284715) and by the Forestry Commission GB. M.D. is supported in part by funding from NERC NCEO and NERC Standard grants NE/N00373X/1 and NE/P011780/1.

Acknowledgements. The following additional people participated in the oak tree leaf measurements: Ian Craig, Steve Coventry, Marc Sayce and David Payne from Forest Research UK; Andrew Burt, Ross Hawton, Jingjing Yan and Meng Yu from University College London; Ewan Pinnington from the University of Reading; and Amy Danson and Jennifer Danson.

Endnote

[1] Otherwise you would have to write a separate intersection detection function for each leaf and block type pair.

References

1. Casella E, Sinoquet H. 2007 Botanical determinants of foliage clumping and light interception in two-year-old coppice poplar canopies: assessment from 3-D plant mock-ups. *Ann. For. Sci.* **64**, 395–404. (doi:10.1051/forest:2007016)
2. Newnham GJ, Armston JD, Calders K, Disney MI, Lovell JL, Schaaf CB, Strahler AH, Danson FM. 2015 Terrestrial laser scanning for plot-scale forest measurement. *Curr. Forestry Rep.* **1**, 239–251. (doi:10.1007/s40725-015-0025-5)
3. Woodgate W *et al.* 2015 An improved theoretical model of canopy gap probability for Leaf Area Index estimation in woody ecosystems. *For. Ecol. Manage.* **358**, 303–320. (doi:10.1016/j.foreco.2015.09.030)
4. Ross J. 1981 *The radiation regime and architecture of plant stands*. Tasks for Vegetation Science 3. The Hague, The Netherlands: Springer. See http://doi.org/10.1007/978-94-009-8647-3.
5. Raumonen P, Kaasalainen M, Åkerblom M, Kaasalainen S, Kaartinen H, Vastaranta M, Holopainen M, Disney M, Lewis P. 2013 Fast automatic precision tree models from terrestrial laser scanner data. *Remote. Sens. (Basel)* **5**, 491–520. (doi:10.3390/rs5020491)
6. Hackenberg J, Spiecker H, Calders K, Disney M, Raumonen P. 2015 Simpletree—an efficient open source tool to build tree models from TLS clouds. *Forests* **6**, 4245–4294. (doi:10.3390/f6114245)
7. Grau E, Durrieu S, Fournier R, Gastellu-Etchegorry JP, Yin T. 2017 Estimation of 3D vegetation density with terrestrial laser scanning data using voxels. A sensitivity analysis of influencing parameters. *Remote Sens. Environ.* **191**, 373–388. (doi:10.1016/j.rse.2017.01.032)
8. Disney MI, Boni Vicari M, Burt A, Calders K, Lewis SL, Raumonen P, Wilkes P, 2018 Weighing trees with lasers: advances, challenges and opportunities. *Interface Focus* **8**, 20170048. (doi:10.1098/rsfs.2017.0048)
9. Béland M, Baldocchi DD, Widlowski JL, Fournier RA, Verstraete MM. 2014 On seeing the wood from the leaves and the role of voxel size in determining leaf area distribution of forests with terrestrial LiDAR. *Agric. Forest Meteorol.* **184**, 82–97. (doi:10.1016/j.agrformet.2013.09.005)
10. Ma L, Zheng G, Eitel JUH, Moskal LM, He W, Huang H. 2016 Improved salient feature-based approach for automatically separating photosynthetic and nonphotosynthetic components within terrestrial lidar point cloud data of forest canopies. *IEEE Trans. Geosci. Remote Sens.* **54**, 679–696. (doi:10.1109/TGRS.2015.2459716)
11. Casella E, Sinoquet H. 2003 A method for describing the canopy architecture of coppice poplar with allometric relationships. *Tree. Physiol.* **23**, 1153–1170. (doi:10.1093/treephys/23.17.1153)
12. Béland M, Widlowski JL, Fournier RA, Côté JF, Verstraete MM. 2011 Estimating leaf area distribution in savanna trees from terrestrial LiDAR measurements. *Agric. Forest Meteorol.* **151**, 1252–1266. (doi:10.1016/j.agrformet.2011.05.004)
13. Zheng G, Moskal LM. 2012 Leaf orientation retrieval from terrestrial laser scanning (TLS) data. *IEEE Trans. Geosci. Remote Sens.* **50**, 3970–3979. (doi:10.1109/TGRS.2012.2188533)
14. Bailey BN, Mahaffee WF. 2017 Rapid measurement of the three-dimensional distribution of leaf orientation and the leaf angle probability density function using terrestrial LiDAR scanning. *Remote Sens. Environ.* **194**, 63–76. (doi:10.1016/j.rse.2017.03.011)
15. Hétroy-Wheeler F, Casella E, Boltcheva D. 2016 Segmentation of tree seedling point clouds into elementary units. *Int. J. Remote Sens.* **37**, 2881–2907. (doi:10.1080/01431161.2016.1190988)
16. Åkerblom M, Raumonen P, Kaasalainen M, Casella E. 2015 Analysis of geometric primitives in quantitative structure models of tree stems. *Remote Sens. (Basel)* **7**, 4581–4603. (doi:10.3390/rs70404581)
17. Clawges R, Vierling L, Calhoon M, Toomey M. 2007 Use of a ground-based scanning LiDAR for estimation of biophysical properties of western larch (Larix occidentalis). *Int. J. Remote Sens.* **28**, 4331–4344. (doi:10.1080/01431160701243460)
18. Calders K *et al.* 2015 Nondestructive estimates of above-ground biomass using terrestrial laser scanning. *Methods Ecol. Evol.* **6**, 198–208. (doi:10.1111/2041-210X.12301)
19. Perttunen J, Sievänen R, Nikinmaa E. 1998 LIGNUM: a model combining the structure and the functioning of trees. *Ecol. Modell.* **108**, 189–198. (doi:10.1016/S0304-3800(98)00028-3)
20. Disney M, Lewis P, Saich P. 2006 3D modelling of forest canopy structure for remote sensing simulations in the optical and microwave domains. *Remote. Sens. Environ.* **100**, 114–132. (doi:10.1016/j.rse.2005.10.003)
21. Cannell MGR, Bowler KC. 1978 Phyllotactic arrangements of needles on elongating conifer shoots: a computer simulation. *Can. J. Forest Res.* **8**, 138–141. (doi:10.1139/x78-022)
22. Åkerblom M. 2017 QSM-FaNNI Matlab implementation source code. See http://www.https.com//doi.org/10.5281/zenodo.800496.
23. Niklas KJ. 1988 The role of phyllotatic pattern as a ‘developmental constraint’ on the interception of light by leaf surfaces. *Evolution* **42**, 1–16. (doi:10.2307/2409111)
24. Potapov I, Järvenpää M, Åkerblom M, Raumonen P, Kaasalainen M. 2016 Data-based stochastic modeling of tree growth and structure formation. *Silva Fennica* **50**, 1413. (doi:10.14214/sf.1413)

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland

ISBN 978-952-15-4140-7
ISSN 1459-2045