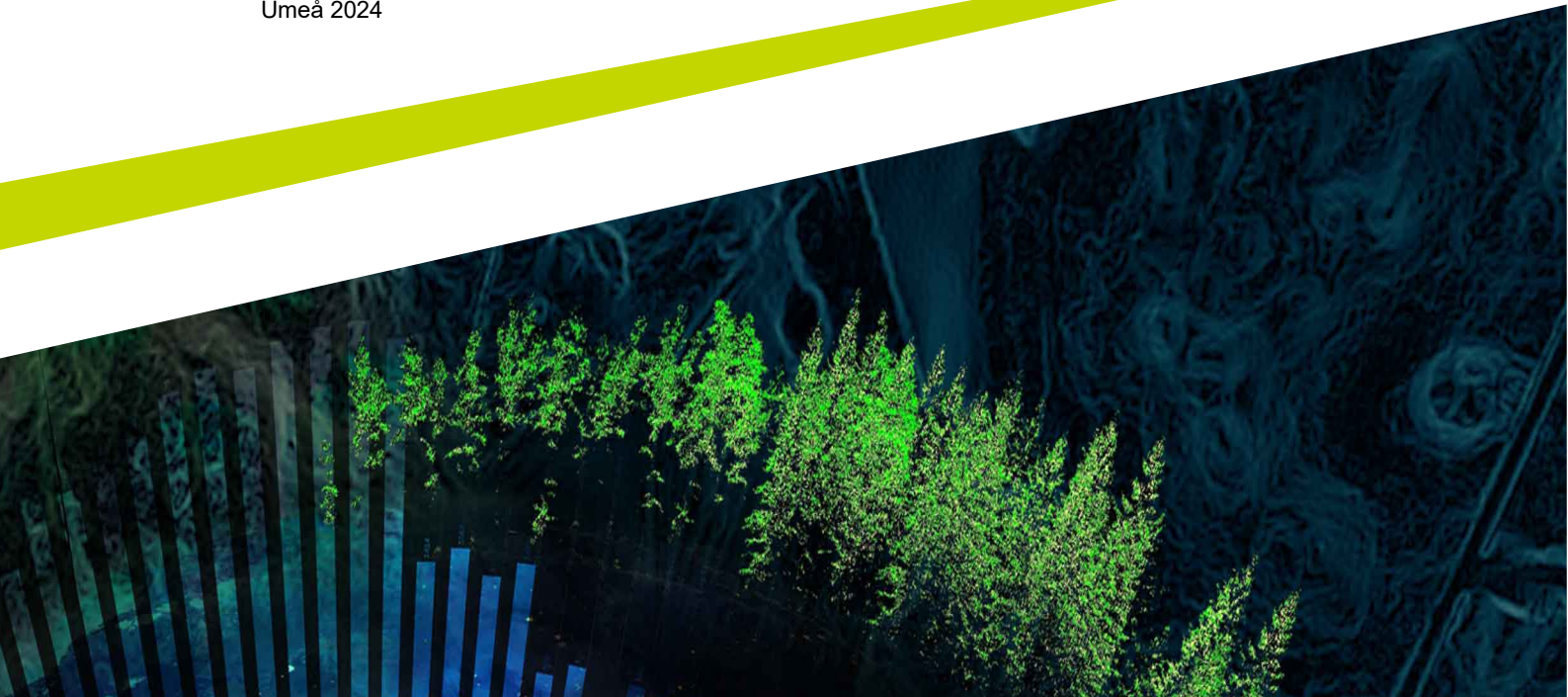
# ForesTRACE.jl: Raytracing Analysis for Utilising Lost Information in Laser Scanning Methods

Jan Zrnovský

Degree project • 30 credits
Swedish University of Agricultural Sciences, SLU
Faculty of Forest Sciences / Department of Forest Resource Management
Forest Ecology and Sustainable Management / Master's Programme
Arbetsrapport / Sveriges lantbruksuniversitet, Institutionen för skoglig resurshushållning,563
ISSN 1401-1204
Umeå 2024



# ForesTRACE.jl: Raytracing Analysis for Utilising Lost Information in Laser Scanning Methods

Jan Zrnovský

| | |
|---|---|
| **Supervisor:** | **Ruben Valbuena, SLU, Forest Resource Management** |
| **Assistant supervisor:** | Cameron Pellett, SLU, Forest Resource Management |
| **Examiner:** | Jonas Bohlin, SLU, Forest Resource Management |

| | |
|---|---|
| **Credits:** | 30 credits |
| **Level:** | Second cycle, A2E |
| **Course title:** | Master's thesis in Forest Science |
| **Course code:** | EX0965 |
| **Programme/education:** | Forest Ecology and Sustainable Management |
| **Course coordinating dept:** | Forest Resource Management |
| **Place of publication:** | Umeå |
| **Year of publication:** | 2024 |
| **Copyright:** | All featured images are used with permission from the copyright owner. |
| **Serietitel:** | Arbetsrapport / Sveriges lantbruksuniversitet, Institutionen för skoglig resurshushållning |
| **Delnummer i serien:** | 563 |
| **ISSN:** | 1401-1204 |

**Keywords:** LiDAR, MLS, occlusion, raytracing, Julia

**Swedish University of Agricultural Sciences**
Faculty of Forest Sciences
Department of Forest Resource Management
Division of Forest Remote Sensing

## Abstract

The structure of a forest ecosystem is an important ecological, environmental, and socio-economic driver. Forest remote sensing is witnessing a rapid development of new technologies and methodologies for quantifying forest structure; however, they are not without flaws. Laser scanning, as one of the major data collection methods in RS, is negatively affected by both the scanned environment and the human error. This master's thesis aims to address and quantify two of the established drawbacks of LiDAR scanning - occlusion, and user focus bias, to further help with LS optimalisation.

A new voxel-based raytracing package for the programming language Julia was developed to analyse occlusion, user focus bias and openness of point-clouds. It merges the start and endpoints of laser beams to construct laser rays, which traverse the voxel environment. In contrast to traditional point cloud analysis, which considers only the endpoints of the beams, introducing ambiguity into derived metrics and losing parts of the scanned information, our raytracing method provides a more comprehensive approach. The introduced raytracing method was demonstrated on the P-TLS platform. With the raytracing analysis performed on five demonstrational scans, we were able to illustrate the effect of user focus bias on occlusion and derived tree height. Additionally, we proposed a new method of measuring tree height using the openness quantifier – the G-T method. In our analysis, the traditional method of deriving tree height using a percentile of a quantile (P90 and P95) greatly underestimated the presumed height, as well as showed susceptibility to the user focus bias quantifier, unlike the more robust G-T method.

*Keywords:* LiDAR, MLS, occlusion, raytracing, Julia

# Table of contents

# List of figures

# Abbreviations

| | |
|---|---|
| ALS | Airborne Laser Scanning |
| DBH | Diameter at Breast Height |
| G-T | Ground-TopVoxel |
| GNSS | Global Navigational Satellite System |
| H-MLS | Handheld Mobile Laser Scanning |
| H-PLS | Handheld Personal Mobile Laser Scanning |
| IMU | Inertial Measurement Unit |
| LAI/D | Leaf Area Index/Density |
| LiDAR | Light Detection and Ranging |
| M-TLS | Multi-Scanning TLS |
| NFI | National Forest Inventory |
| PAD | Plant Area Density |
| P-MLS | Personal Mobile Laser Scanning |
| RTM | Radiative Transfer Model |
| RS | Remote Sensing |
| SLU | Swedish University of Agricultural Sciences |
| S-TLS | Single-scan Terrestrial Laser Scanning |
| TLS | Terrestrial Laser Scanning |
| UAV | Unmanned Aerial Vehicle |
| UAV-LS | UAV-based Laser Scanning |

# 1. Introduction

Forest owners and managers were always interested in the state of their property, be it the well-being of their game to hunt or the quality of the wood to build ships from for the medieval royalty, or the number of trees to thin and to sell on the market at the right price, for the forest companies nowadays. As time passed, the information needs of the forest owners and companies became increasingly complex with specific niches (Tomppo et al., 2010), like crown density (Solberg & Strand, 1999), soil composition (Hanberry et al., 2012), or ecological microhabitats (Larrieu et al., 2018) – thus becoming challenging for the surveyors. In general, the needs of the managers on the forest plot information should be as precise as possible and up to date, and need to accurately describe the stand qualitative and quantitative state, such as wood supply and species composition (White et al., 2016). Acquiring all necessary information about forest stands has been and still is a labor- and time-intensive task. With the rise of accurate statistical methods, the forestry community saw the rise of National Forest Inventorying (NFI) in many states around the world, with Fennoscandian countries being the first (Tomppo et al., 2010). These NFIs rely on in situ sample acquiring methods, meaning a survey team would go to the forest stand and retrieve the measurable metrics on forest sample plots. Tree attributes like stem diameter (DBH – diameter at breast height), tree species composition and tree height are measurable in the forest plot by non-destructive means and can be used to further derive other tree and stand metrics by the needs of the consumer (Wang et al., 2019). Tree height has been measured by either destructive or non-destructive means. While destructive methods are acceptable while harvesting trees, they are not acceptable during the pre-harvest period. When the tree is felled, it is possible to physically measure the length between the tree base and its top, thus resulting in a precise measurement. The non-destructive methods have usually been carried out using a hypsometer or a similar device based on the triangle similarity, thus bringing some accuracy errors (Andersen et al., 2006). Measuring every single tree in the whole country is an unimaginable task but thanks to sampling methods the forest management have been able to predict the actual state of the land. However, sample plots can be placed in remote areas and far from roads, thus the sampling method carries the burden of higher prices and long time to competition. Most NFIs have been carried out once every 5 to 10

years, depending on the country and the desired information about the environment (Tomppo et al., 2010).

As a response to the time and cost inefficiency of in situ measurements, both forest researchers and managers have been looking for a way to ease the hard work and increase the temporal resolution through new technology (Balenović et al., 2020). One of the first ideas to use remote sensing in forestry was in the 1950s, when forest scientist in Central Europe discussed if aerial black-and-white photographs of forest stands could be a useful tool in forestry practice, but there was a lack of interest from forest managers for data acquired in this way (Fassnacht et al., 2023). However, in 1987 aerial image interpretation methods were already well established and used as support information for forest managers while management planning (Fassnacht et al., 2023). Thanks to further technological advancement and newly introduced remote sensing methods, the forestry community has been able to start quantifying both horizontal and vertical forest structure. This task is relatively difficult, and in some cases impossible, with traditional in situ surveys.

It was proven, that vertical and horizontal forest structure, as well as the stand species composition, influence local- and micro-climate surrounding the forest stand by working as a wind dampener and a temperature buffer, as well as moisture reservoir (De Frenne et al., 2021; Zellweger et al., 2019). There is also the argument that abrupt changes in the forest ecosystems can lead to change in diversity, available niches, and nutrient availability (Von Arx et al., 2013). The ability to quantify, model, and predict forest structure and its changes allows for a comprehensive understanding of forest biomass, carbon sequestration, storage properties, and their responses to disturbances. As such, forest managers and owners should strive for healthy ecosystems through sustainable management to not only promote quality timber growth, but also to help with climate change mitigation. As it was proven, forest ecosystems affect the temperature cycles of the stand surroundings, subsequently the macroclimate, with their diverse structure and energy transfer cycles (Aussenac, 2000; Ligot et al., 2014). Thus, it is clear that the forest research and practice community should be interested in documenting not only the current state of the forest structure, but also the changes in it, and what benefits it could have for the global climate change mitigation.

This is where remote sensing methods come into the spotlight, as they enable the monitoring of ecosystem changes at various scales, ranging from large ecosystems, such as the deforestation state in the Amazon rainforest, to the smallest scales. With the ever-increasing resolution and fast processing of acquired data using remote sensing methods, the remote sensing community is able to document the three-dimensional forest structure, with high temporal resolution, like never before (Valbuena et al., 2020). Light Detection And Ranging (LiDAR) it is one of the

leading data acquiring methods in remote sensing. It emits either near infrared, green or SWIR laser beams into an environment and uses sensors to detect returned beams that reflected from objects in the scanned area (Wang et al., 2019). Results of Laser Scanning (LS) are high-precision three-dimensional point clouds of data, based on the returns range and orientation to the scanner (Li et al., 2012) or with the use of LiDAR full waveform systems, the results are datasets also containing the backscatter distortion of the each returned laser ray (Mallet & Bretar, 2009). Forest scientists started adopting simplest forms of laser scanning in the 1990s (Bauwens et al., 2016). Publications typically classify LS data collection in forestry according to the platform it is mounted to (Bauwens et al., 2016; Calders et al., 2020). Covering largest areas is the space-born platform on satellites and the ISS (space shuttles historically), with its advantage of continuous scanning and creating long time series comes faltering spatial resolution in comparison to other platforms. Most used platform nowadays is the air-borne one, where the scanner is mounted to an airplane or an unmanned aerial vehicle (UAV). Third classification is the terrestrial platform, which can be further subdivided into "stationary" and "mobile". All three platforms find their adequate uses in forest inventorying and see continuous interest from the scientific community and forest practice. Thanks to the mentioned interest from the community, their technology continues to be developed to deliver ever higher spatial, spectral, and temporal resolution (Bauwens et al., 2016; Calders et al., 2020; White et al., 2016).

An airborne LiDAR platform carried by an airplane or a helicopter is classified as an Aerial Laser Scanner (further just ALS), and UAV-based Laser Scanning when mounted to a drone (further just UAV-LS). Both ALS and UAV-LS create high density point-clouds by combining return data with an Inertial Measurement Unit (IMU) and a Global Navigational Satellite System (GNSS), but they operate on different flight levels. The flight altitude of ALS in forest inventorying typically ranges between 0.5 and 3 km (Goodwin et al., 2006; Næsset, 2009) and can reach surface height measurement accuracy of less than a meter (Næsset, 2009; Næsset & Bjerknes, 2001). Inventorying with the UAV-LS is typically performed no more than 150 - 300 meters above the ground, depending on the law regulations of the country where the survey takes place (E. Hyyppä et al., 2020; Puliti et al., 2015). The mentioned difference in flight altitude of ALS and UAV-LS, as well as the flight speed and path overlap, results in differing spatial resolution, as the laser beams footprint is not infinitely small but cone-shaped. Due to the distance difference, the ALS LiDAR footprint covers a larger area on the ground, in comparison to the smaller footprint UAV-LS in lower flight altitudes (Luo et al., 2023). Advantage of ALS, in comparison to TLS, is the aforementioned use of precise and continuous GNSS geo-location, which is possible due good satellite signal coverage in the flight altitudes (Sferlazza et al., 2022). Such ALS point-clouds represent spatial distribution of elements in the observed canopy and are

used to generate terrain maps and both stand and single tree attributes (Bauwens et al., 2016; Hyyppä et al., 2008). ALS and UAV-LS can observe large areas efficiently; however, this ability is balanced by lower spatial resolution. With less return points per $m^2$ the ability of ALS to sufficiently describe the ground vegetation is reduced, thus the information about the understory is to some extent obscured (White et al., 2016).

To study ground vegetation in the shrub layer, bushes and lower parts of the canopy, Terrestrial Laser Scanning (TLS) takes over the data extraction by scanning from below the canopy, inside of the forest stand. TLS, in comparison to ALS, is capable of scanning with millimetre-level precision, thus deriving tree metrics such as DBH, timber volume, stem curve or Leaf Area Index/Density (LAI/D) or others is possible (Bauwens et al., 2016; L. Li et al., 2021; W. Li et al., 2012; Wang et al., 2019). Approaches to estimating tree height from point clouds differ in the forest research community. Two of the most used methods either measure the lowest and highest point of the point cloud (Calders et al., 2015; Saarinen et al., 2017) or extract the height using a point distribution quantile. Commonly used percentiles of a quantile to determine tree height are 90th, 95th or 99th (Mao et al., 2019; Næsset & Bjerknes, 2001; Stovall et al., 2017). However, tree height estimation while remote sensing has been proven to generally underestimate the actual height if compared to destructive methods (García et al., 2011). This phenomenon of tree height underestimation while using ground-based methods is explained as the effect of lower canopy branches occluding the top parts and having higher density of points (Bauwens et al., 2016; García et al., 2011).

At first, Stationary Single-scan Terrestrial Laser Scans (S-TLS) have been carried out, but it was proven that they deliver lacking data due to occlusion effects, as seen from Figure 1. To battle occlusion, co-registered Multi-scan Terrestrial Laser Scanning (M-TLS) or Mobile Laser Scanning (MLS) methods were presented (Bauwens et al., 2016). M-TLS uses the same statically standing scanner, but the observed area is captured multiple times, from within and outside the plot, to ensure low occlusion ad hight point-cloud overlay. This method of M-TLS delivers very dense and precise point clouds; however, it is expensive, time consuming and cumbersome. Another branch of TLS: MLS, tries to overcome beforementioned problems – occlusion effects and time demands. It does so by mounting a scanner on a moving medium, be it a roller on tracks, car, or a person. In a forest ecosystem setting cars or ATVs (all-terrain vehicles) are bound either on roads or into sparse forests, thus the most progress in the past years has been made with Personal Mobile Laser Scanners (P-MLS), especially Handheld Personal Mobile Laser Scanners (H-MLS or H-PLS) (Balenović et al., 2020; Bauwens et al., 2016). With the H-PLS method of data acquisition, the operator holds the scanner in their hand and walks inside the forest while continuously scanning. Thus, has a relatively free range of

movement to scan, even in a dense environment (Bauwens et al., 2016; Calders et al., 2020). While performing ground-based LS, survey crews place multiple spheres on the ground and record their GNSS coordinates, for co-registering and georeferencing the point clouds during the post-processing. Although the operator can move relatively freely, the GNSS signal is degraded by the tree canopy layer and difficult terrain where forest ecosystems usually grow, thus making geo-location harder (Sferlazza et al., 2022). Movement of the operator also negatively affects the IMU sensors, making the point-clouds less accurate (Liang et al., 2014). While comparing TLS and ALS, it is essential to discuss other trade-offs of TLS. The main downsides of TLS and MLS are mainly the cost and time ineffectiveness. When a drone or a plane perform laser scanning, it generates a point cloud covering the entire observed region. In contrast, ground-based LiDAR point clouds are smaller and are typically utilized in a sample plot design. Both line-of-sight and footprint size significantly impact the precision of TLS and MLS, especially in dense forest environments where the LiDAR beam's effectiveness diminishes beyond 100 meters (Calders et al., 2020). Numerous remote sensing inventorying papers have presented methodologies that combine ALS and TLS to produce highly accurate wall-to-wall maps for expansive areas (White et al., 2016).

When it comes to remote sensing methods, one must take into consideration the shortcomings of the technology used. LiDAR has been proven many times to be a powerful tool but some of its downsides have been described by literature. Probably the key negative factor of all beforementioned LiDAR platforms is the possibility of occlusion (Abegg et al., 2017; Balenović et al., 2020; Bauwens et al., 2016; Jurjević et al., 2020; L. Li et al., 2021; Mathes et al., 2023). Occlusion, the opposite of inclusion, occurs when an object is unintentionally hidden behind another one while scanning. And thus, some information could be missing in the finished point-cloud – which can carry misleading information and uncertainty in measurements and derived metrics. Occlusion is primarily influenced by the scanned environment, weather, and to some extent by human error. When the scanned forest plot is either dense, has a thick canopy, or an abundant understory growth, we can expect higher levels of occlusion compared to sparse open forests (Balenović et al., 2020; Bauwens et al., 2016; Jurjević et al., 2020). Fog or rain can also skew the scanning result by reflecting laser beams (Abegg et al., 2017). Human error steps into the equation if a scanner is nor properly set up or the location of the scanner (flight or walking path for ALS/MLS respectively) is not chosen properly (Abegg et al., 2017; Bauwens et al., 2016). Some efforts to address MLS occlusion were undertaken, as seen for example in the work by L. Li et al. (2021). In their trial, the team created a new algorithm to choose scanning spots in a plot, so the final occlusion rate is minimal. However, the premise for their method is that the location has been scanned before and we know the object distribution (trees and stones) in

the forest plot. However, for other laser scanning platforms, efforts to minimize occlusion are, so far, relatively lacking.

Part of the occlusion minimizing process is its quantification, which has been done by several studies, either on a 2D plane using pixelized rasters or in 3D through voxelization (Abegg et al., 2017; Béland et al., 2011; L. Li et al., 2021; Zong et al., 2021). The primary concept behind voxelization is to populate the scanned space with elements - voxels (usually same sized cubes) and determine the attributes of these voxels based on the points within them. Point cloud voxelization was at first used to quantify volumetric attributes, such as Plant Area Density or LAI/D (Pimont et al., 2018) and light availability (Stark et al., 2012) as well as timber volume (Bienert et al., 2014) and how are these results affected by LiDAR properties (Almeida et al., 2019). However, the mentioned applications rarely involved raytracing, and only analysed endpoints encompassed in the voxels. The voxel based non-raytracing methods usually divide the voxels discretely as “filled” and “empty”, based on the number of points it contains, while losing the information about the position and orientation of the scanner. Therefore, not taking into consideration possible occlusion. As mentioned before, if occlusion occurs it can skew the resulting voxel attributes. A voxel might have had some informational value but was hidden from sight of the scanner and might end up bearing no information in the end product. Subsequently, voxelization approach has found application in ray tracing methods as well (Schneider et al., 2019). The ray tracing approach enhances this simple delineation by providing information about the trajectory of the laser beam and what voxels it traversed before hitting the return point. Thus, it can be determined if a voxel was scanned or occluded from sight of the scanner, something the non-raytracing method cannot determine with certainty. First uses of raytracing in forest research were done to assess the leaf attributes of trees and subsequently stands, as demonstrated by Bittner et al. (2012) and Morsdorf et al. (2007). This approach used the Mone-Carlo raytracing method to determine reflective properties, to be used in a Radiative Transfer Model (RTM). RTMs are models used to predicting the relationships between stand structure and radiation cycles happening within the stand (Ligot et al., 2014). Later raytracing research papers used the voxel (or pixel) traversal approach to determine occlusion in a similar fashion to the approach we decide to take (Abegg et al., 2017; Schneider et al., 2019). Abbegg et al. (2017) searched for the relationship between occlusion and TLS placement in simulated forest stands. Their raytracing approach was done on a 2D plane, and the analysis was done both for single and multi-scan TLS. Schneider et al. (2019) used combined above canopy scanning from a crane with under canopy TLS to determine the occlusion rates inside of the canopy using voxelization and raytracing traversal algorithms.

Although all methods of LiDAR scanning are continuously being debated and developed by the RS community, not many studies have been made to address how to mitigate point cloud biases, such as occlusion or human operator error in H-MLS. In this thesis, we present an idea with some suggested methods on how to quantify these errors via raytracing, to help streamline LiDAR data acquisition in the future. Aim of this study is to:

- Create an open-source raytracing package for the RS community.
- Demonstrate LiDAR errors of Occlusion rate and User focus bias using the introduced methodology.
- Suggest a new method of calculating tree height using acquired raytracing metrics.

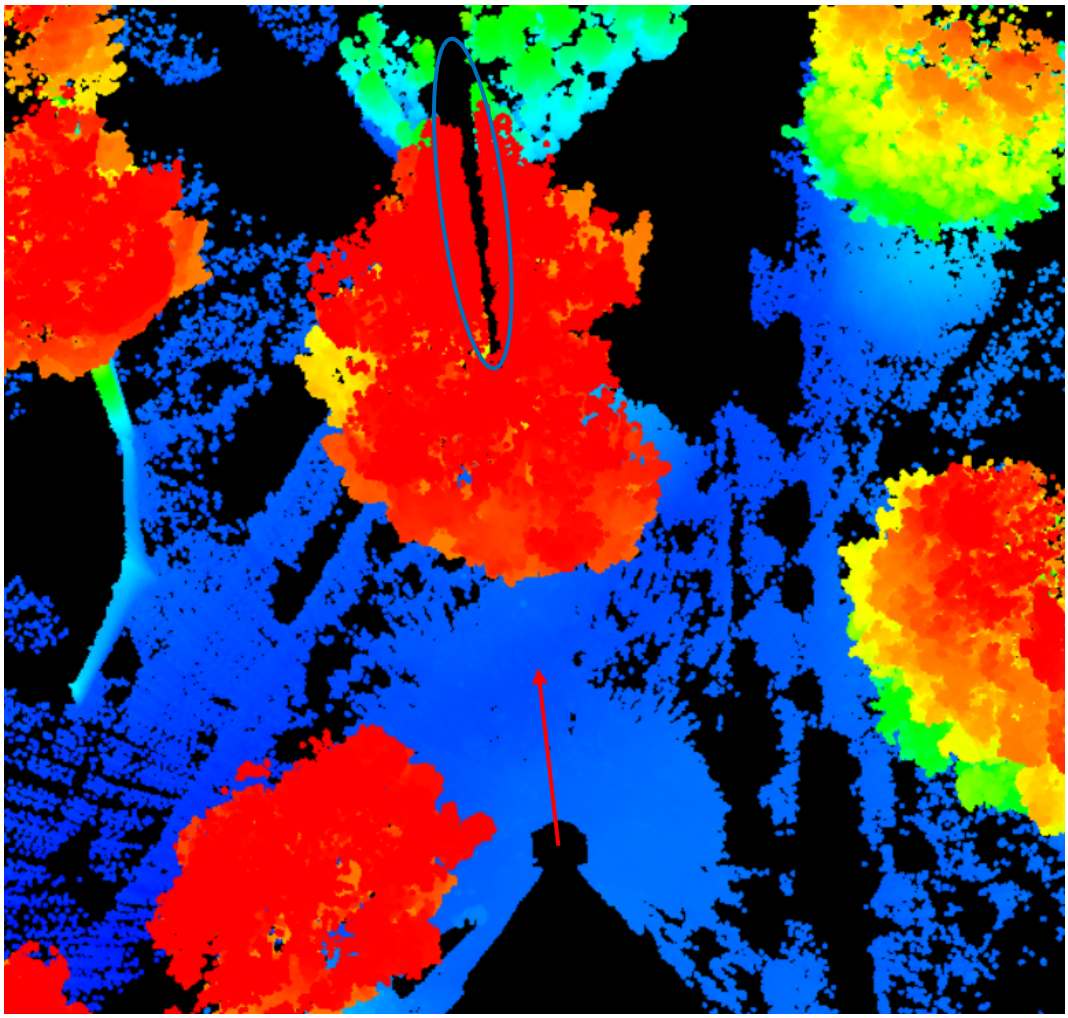


*Figure 1. Occlusion effect happening while performing a single scan TLS. An entire section of a tree, represented by the blue oval, is occluded by a trunk of another tree. The red arrow represents a beam direction from the scanner. Scan "static".*

# 2. Methodology

## 2.1 Data Acquisition

### 2.1.1 Used instrument - GeoSLAM THLS

For acquiring the LiDAR point-clouds analysed in this thesis we used the H-PLS GeoSLAM ZEB Horizon (see Figure 2). Technical specifications: laser wavelength 903 nm, maximum range 100 m, scan rate 300 000/s (16 lines @ 10 GHz), beam divergence 3.0 mrad. We chose the H-PLS platform, as it has received a lot of attention in recent remote sensing studies, for its novelty and great potential in forest inventorying.

The 3D point information is created by spinning its 2D time-of flight (TOF) laser range scanner while connected to an inertial measurement unit (IMU) on its motor. Instead of the need to geo-reference or add co-registering points into the scanned plot a simultaneous localization and mapping (SLAM) algorithm accurately creates the resulting point-cloud from the IMU and TOF data. The SLAM algorithm relies on objects and features being inside of the scanned area, not on GNSS data, which are often inaccurate in forests and other dense environments (Balenović et al., 2020). The algorithm co-registers the map (or the point-cloud) based on scanning these objects repeatably. Thus, it is important for the operator to create loop closures while scanning the area. In our case, at least one loop closure was ensured by starting and ending the scanning process at the exact same spot. To further help the algorithm, walking in circular pattern and regularly aiming the scanner back at what has already been scanned was also done according to the recommendation of the manufacturer.

The collected raw data by the scanner needed to be processed by a GeoSLAM Hub processing software before the analysis. The software is fed raw mapped data in a .geoslam format and the processing results in two files needed for the ray tracing analysis – first the .laz file which contains data about end points of the point cloud, and second a .traj-gs file with data about scanners position (rays beginning).


*Figure 2. GeoSLAM ZEB Horizons scanning head with a handle and a data logger.*

### 2.1.2 The trial site

The demonstrational trees used in the project are two well-established pines (*Pinus sylvesteris*) in a city-park setting in proximity of the Umeå (Sweden) SLU building – 63.81N, 20.31 (WGS84). These two trees were chosen for their height, which is tall enough to mimic trees in a mature forest stand, as well as for their accessibility for repeated scanning (see Figure 3). The surrounding of the trees, given its park structure, was open and thus the operator had free range of movement during the demonstrational scanning, as well as clear view of the treetops for the scanner to register them, and for measuring the tree height using a hypsometer for reference. The pines stand close to each other and given the open structure in their surroundings, they would be easily spotted in the point cloud during the segmentation. GNSS positioning spheres were not placed in the scanned area, as geo-referencing was not necessary in this study.

We measured tree height in situ with a hypsometer Haglöf EC II-D five times for both pines respectively, to later average the values and use it as a reference to LiDAR height measurement.

*Figure 3. Pine A (background) and Pine B (foreground).*

### 2.1.3 Demonstrational scans

To demonstrate user bias (focus) and its effects on scan quality and occlusion, we made five scans of the two pines (see Figure 4). All five scans had the same starting and end point – about 10 meters from pine B. The scans were done as follows:

- Scan 1 “static” – simulated a stationary TLS, scanner was not moved and was left running for roughly 30 seconds.

- Scan 2 “line” – the operator walked in a straight-line diagonal to the tree axis for 20 meters and back to the starting point, turning in the middle to ensure a good loop cycle for the SLAM algorithm. The operator did not aim the scanner at the trees but in the direction in which they walked.

- Scan 3 “bottoms” – The operator intentionally looked at the base of the tree trunks and walked close to them, under the canopy. This scan is supposed to simulate operators who would not scan the crown area properly, thus the

point density is to be higher for lower parts of the canopy and the tree trunks, than for the top of the crowns.

- Scan 4 “tops” – In this scan, the operator was intentionally paying more attention to the treetops than in the previous scans. They walked further away from the trees in a circle. In comparison to scan C, this scan was meant to emphasise the opposite, thus in the end create more balanced point density for both the crowns and the rest of the trees.

- Scan 5 “control” – Last scan was created as a control for the other scans, as the operator did not intentionally make any mistakes, walked both close and far around the trees, as well as in the middle of them, aiming at both tree trunks and crowns.

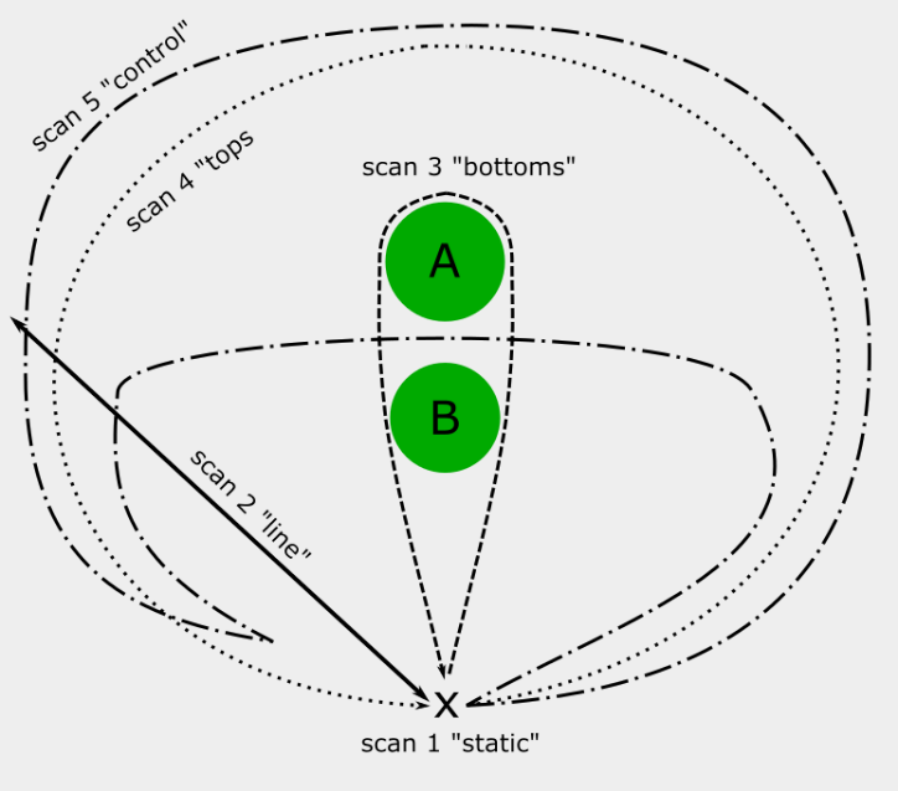



*Figure 4. Schematized paths taken during the five scans we took. X in the bottom represents the start and end position of the scanner during all five scans. Green circles A and B represent the two scanned pines.*

## 2.2 Data Analysis using the traditional point cloud method

Traditional point cloud analysis for all five scans was done using the lidR package (Roussel et al., 2020) for the programming language “R”, with which we segmented individual trees, extracted pines A and B and derived height metrics from them, we also classified ground points for later use.

The individual tree segmentation started by creating a Canopy Height Model and rasterizing the point cloud, using the rasterize_canopy() function with arguments rasterize_canopy(las = las, res = 0.5, algorithm = p2r(0.15)). The rasterized canopy was smoothed out using a median filter with a 3x3 kernel. To locate trees themselves we used the locate_trees() function, with arguments locate_trees(las = schm, lmf(4, hmin = 15, shape = ("circular"))), which identifies possible tree. The segmentation was performed using the segment_trees() function, with the Dalponte algorithm - segment_trees(las = las, algorithm = dalponte2016(chm = schm, treetops = ttops)).

After the individual trees had their IDs, we could manually extract pine A and B from the point cloud into their separate .laz files. Before deriving the tree height metrics, we have filtered out the ground points, to reduce the effect these points would have on skewing of the results. Filtering out the ground points was done using the filter_poi(las, Classification != LASGROUND) function.

To extract tree metrics, we deemed important for this project, we used another package from the lidR environment called “lidRmetrics”. The metrics extracted for both pines were zmax, zq95 and zq90. Zmax indicates the height of the highest point in a segmented point cloud. Zq95 and zq90 refer to the height of the $95^{th}$ and $90^{th}$ percent of the height quantile within the segmented point cloud.

## 2.3 Raytracing method

The goal of the presented method is taking into consideration not only the end points of the laser point-cloud, which can report skewed data due to occlusion and operator bias, but to create and analyse a voxel environment and laser rays traversing it, through ray tracing. Ray tracing sees the ray as a line segment with a starting point (scanner head) and an end point (object from which the laser returned from). These segments traverse a three-dimensional space made with cubic voxels. While performing the analysis we generate the rays and test how they interfere with the voxels in the environment. For each individual voxel we check if rays passed through it, stopped in it, or did not interact at all. At the end of the analysis, the 3D

environment of voxels tells us not only in which voxel how many laser beams ended but also through which voxels the beams have passed and what voxels were not transacted by any laser rays. With this information, we can quantify and visualize the empty space, occlusion, and user bias, perhaps more.

### 2.3.1 Julia Programming language

For writing the code needed to run our ray tracing analysis efficiently, we chose the Julia programming language (Bezanson et al., 2017). Julia was developed under the MIT license and is meant to be both high-level and fast language for general use, however it is now used mostly for numerical analysis and computational science. It is regarded as an easy-to-read programming language for humans, similarly to Python. Being released in 2012, Julia is a relatively new coding language in the open-source world, but there are many high-level packages created for various uses already. We took advantage of the Meshes.jl package, which helps to create 2D and 3D geometrical objects in space, as well as LazIO.jl for laser data manipulation and DataFrames.jl (Bouchet-Valat & Kamiński, 2023) with SortMerge.jl for easy data frame manipulation, as well as Makie.jl (Danisch & Krumbiegel, 2021) for visualization.

### 2.3.2 Creating the LiDAR rays

To be able to perform the ray tracing analysis, we must find and merge corresponding start and end points of the LiDAR rays. Start and end point data files contain large quantity of information about the laser beam, but only two are important in our case. Those are the coordinates of both points and their time stamp. In some applications, laser intensity or other knowledge about the laser ray might be useful, but we decided not to use them. Since the scanner head shoots more than 3,000 rays each second, we cannot simply merge the end and start points based on their row number but using their mentioned timestamp. Due to the speed of light not being infinite the points rarely had an identical timestamp, that is why we chose a merging method with a threshold – 50 milliseconds in this thesis – to find points that should theoretically match. We used a SortMerge.jl packages sortmerge() function to successfully find corresponding trajectory and end points.

To ensure good ray coverage while performing the ray tracing analysis, we added a 5-meter buffer to both sides in the X and Y axis while filtering out the endpoints from the master point cloud. To filter the LiDAR points, we used the introduced filter!() function, which filters only the points contained in a given geometry – the segmented tree plus the buffer in this project. If we took ray end points only from within the segmented tree area, we would introduce artificial errors (faux occlusion), as some rays might have passed through the analysed space but not

ended there. Size of the buffer is something to be tested in other studies, as it could be changed according to the density of the analysed forest plot.

When a data frame of all rays – corresponding starting and end points – from within the segmented tree area with added buffer was generated, we randomly selected 400,000 rays and used them for the ray tracing analysis. The main reason for running the ray tracing analysis with only a random sample of rays is performance, nonetheless we accept that sampling of rays could lead to some inconsistencies if the original data frame from which we sampled is marginally bigger.

### 2.3.3 Voxelization of the environment

First step of ray tracing analysis is creating a three-dimensional space made of voxels – environment.

While taking advantage of Julias structs, we created our own mutable struct called "Voxel", this object bears 3 sets of information:

- Its dimension and location in space in the .poly field. It takes over the characteristics of an object called Box from the Meshes.jl package. The dimensions and location in space are given by a starting point and an end point. The presumption for creating such Voxel is that it will be perpendicular to the Cartesian 3D space with axes X, Y and Z and it will have a given side length.

- Second field of the Voxel struct is .pass, which tells us how many rays passed through this Voxel. In this case passing means the ray could both only pass and continue or pass through the voxel and end in it. If a ray ends in a voxel, it must also pass through it.

- Last field .stop holds the information of how many rays ended in this voxel.

We came up with a function that creates this 3D environment of cubic voxels named create_voxels(), this function takes the extent of the point cloud to be analysed in X, Y, Z axis and our desired voxel side length. Within the function, we take advantage of the Meshes.jl package and create a cartesian grid, to act as a building guide (scaffolding) for the .poly part of the voxels to be made. Result of this function is a vector of Voxels of given locations and uniform side length, with empty .pass and .stop fields to be written into in the latter parts of the analysis. When creating the environment, it is important to make sure both the extent and the desired voxel resolution are in the same units.

For purpose of this study, we decided to run the ray tracing analysis (for both trees in all five scans) in three different resolutions – voxel side lengths of: 50 cm; 25 cm and 10 cm.

### 2.3.4 Raytracing analysis

At first, a function to delete all voxels under presumed ground was implemented - filter_underground_occ(). This function is crucial especially with sloping plots, as the voxel environment is created according to the coordinate extent of the point cloud. Thus, with sloping plot, the environment would contain a considerable number of voxels underground and the raytracing analysis would result in higher occlusion rates. The function firstly groups all the voxels within the environment into columns with the same X and Y coordinates using our get_middles() and DataFrames groupby() functions. The introduced get_middles() helps with grouping voxels into columns, as it extracts the coordinates of each voxels centre. For every column, a kernel filters corresponding ground points using the introduced filter!() function. The size of the filtering kernel was established as 3x3 times the voxel resolution – meaning nine columns. When the ground points are filtered, their mean Z value represents the ground height and all voxels below it, in the particular column, are deleted.

After an artificial 3D environment encompassing the point-cloud was constructed and the underground voxels were filtered out, we could proceed to the ray tracing itself. The raytracing analysis was done by testing intersection of both the trajectories and the return points of the LiDAR rays with the voxels in the created environment. We formulated two new intersect functions, as the computational time of the inbuilt Meshes.jl hasintersect() function was unacceptable.

Both of the new intersect function iterations test how a given geometry interacted with an observed voxel. First iteration tested if a LiDAR ray (represented by a segment geometry) passed through the voxel or not. The segment intersection test was based on the Möller–Trumbore intersection algorithm (Möller & Trumbore, 1997). The segment representing the LiDAR ray is projected on each axis and then is checked for parallelism with the planes defined by the voxel extent. If the plane and the projected segment were parallel, there was no intersection. In other cases, intersect points with the plains were calculated and if they lied in the extent of the voxel, the ray and the voxel intersected. The second iteration checked if an end point (represented by a point geometry) of a given laser ray ended in the observed voxel. Coordinates of the end point were extracted, and a simple evaluation was carried out. A Boolean TRUE value was returned only if the X, Y and Z coordinates lied in-between max and min coordinates of the voxels corresponding axis.

These two intersect functions were wrapped in an umbrella function that cycles all given voxels with a ray vector and a stop point vector. If a ray passed through the observed voxel, a value of 1 was added to the .pass data field and if a ray stops in the voxel, 1 was added to the .stop field. Due to longer calculation time, a progress bar from the macro @progress in TerminalLoggers.jl and ProgressLogging.jl was included to keep track of the process. The result of the raytrace!() function was a modified voxel vector with filled in information in the .pass and .stop fields.

In this thesis, we used the ray tracing information about the voxel intersections to calculate three quantifiers – openness, user bias and occlusion. Openness (Eq. 1) tells us the proportion of ray passes to passes and stops in a voxel, thus assessing how much interference, causing rays to stop, there is in the voxel. If openness of a voxel is 1, then we can presume that all the rays passed a voxel and there was nothing the rays could return from. If on the other hand it results in 0, there is an object from which all the rays bounced from. While working on this project, we labelled voxels as "open" if their openness was higher or equal to 0.95. In the equation $.pass_{Vi}$ represents the number of LiDAR beams passing the given voxel and $.stop_{Vi}$ represents the number of beams returning from the given voxel.

$$Openness_{Vi} = \frac{(.pass_{Vi} - .stop_{Vi})}{.pass_{Vi}} \quad (1)$$

User bias quantifier – Focus (Eq. 2) tells us the proportion of a given voxel .pass to the environments .pass sum, hence quantifying how much attention was given to a given voxel. We could then calculate variability of user bias for the whole system – by calculating standard deviation and variability using the Julia package Statistics.jl. This variability tells the user if the area was scanned evenly or inconsistently. If the standard deviation was high, we can presume the plot was scanned heterogeneously as some voxels received significantly more attention than others. If on the other hand the variability is low, we presume the scanning was done more homogenously.

$$Focus_{Vi} = \frac{.pass_{Vi}}{\sum .pass_{V}} \quad (2)$$

Last thing we checked with the ray tracing results was the occlusion, that is if the voxel had zero rays passing through it. If that was true, the voxel was classified as occluded – hidden from view or unscanned by the operator.

For ease-of-use, we implemented the rt_quantifiers() function which calculates the mentioned quantifiers for each voxel and adds them to the raytraced DataFrame.

An example of a few Voxels after the raytracing analysis can be seen in the Figure 5.

| Row | poly<br>Box… | pass<br>Int64 | stop<br>Int64 | openness<br>Float64 | focus<br>Float64 | occlusion<br>Float64 | middles_x<br>Float64 | middles_y<br>Float64 | middles_z<br>Float64 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Box{3, Float64}(Point(-5.25, 10.… | 60 | 29 | 0.516667 | 7.98796e-6 | 0.0 | -5.375 | 9.875 | 9.625 |
| 2 | Box{3, Float64}(Point(-5.0, 10.0… | 68 | 44 | 0.352941 | 9.05302e-6 | 0.0 | -5.125 | 9.875 | 9.625 |
| 3 | Box{3, Float64}(Point(-4.75, 10.… | 53 | 26 | 0.509434 | 7.05603e-6 | 0.0 | -4.875 | 9.875 | 9.625 |
| 4 | Box{3, Float64}(Point(-4.5, 10.0… | 89 | 53 | 0.404494 | 1.18488e-5 | 0.0 | -4.625 | 9.875 | 9.625 |
| 5 | Box{3, Float64}(Point(-4.25, 10.… | 94 | 6 | 0.93617 | 1.25145e-5 | 0.0 | -4.375 | 9.875 | 9.625 |
| 6 | Box{3, Float64}(Point(-4.0, 10.0… | 121 | 13 | 0.892562 | 1.6109e-5 | 0.0 | -4.125 | 9.875 | 9.625 |

*Figure 5. Example of six voxels after the raytracing analysis was performed, as well as the raytracing quantifiers were added (scan "tops"; voxel resolution 0.25; Pine B).*

To project the representation of the raytraced voxel space, we created four visualization functions - voxel_viz_openness(), voxel_viz_focus(), voxel_viz_solids() and voxel_viz_occlusion(). Iteration of the functions for visualizing the quantifiers were based on the Meshes.jl viz function. Each of the functions returns a GLMakie 3D Scene for visual interpretation. For details See Appendix 1 – list of introduced functions.

### 2.3.5 G-T height method

In this thesis, we present an alternative to the established tree height measurement methods derived from pure point cloud interpretation, with the information we got from the ray tracing analysis. The presented G-T method uses the openness and occlusion metrics we calculated to determine the individual tree height. In this suggested method, the tree height is the distance between the highest voxel of the tree, which is not considered open, and the ground point with the same X and Y coordinates (see Figure 6).

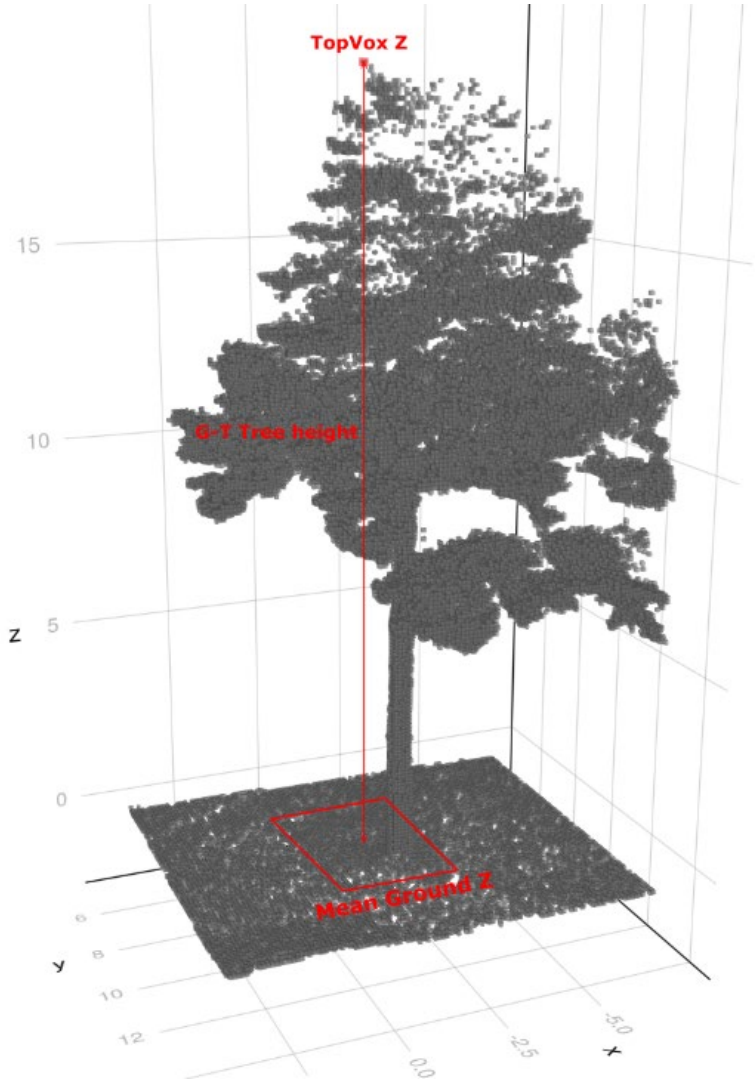



*Figure 6. Illustration of G-T tree height assessment. Tree height is measured as the distance between the highest non-open and non-occluded voxel (TopVoxZ) of the tree and the mean Z value of a 2x2 m rectangle on the ground below it (GroundZ).*

To find the highest voxel of the tree, post ray tracing voxels within the environment (a segmented single tree) were firstly grouped into columns of equal X and Y coordinates but changing Z coordinates, as done before for the filter_underground_occ() function. Then we took each column separately and found its highest non-occluded and non-open voxel, using a top_vox() function. For the voxel to be considered not open, its openness must have been lower than 0.95. As a failsafe to eliminate scanner errors and outliers, the chosen voxel also must have been hit by at least two laser beams – its .stop value must have been greater than one. Then using get_topvox() function, the initial topvox() function was cycled through all of the voxel columns and detected the global maximum. The get_topvox() function returns the X, Y and Z coordinates of the highest non-open voxel of the segmented tree - TopVox.

After the TopVox coordinates were known, we measured the length between its centre and the ground perpendicular to it (Eq. 3). Using the lidR package for R, we classified ground points with the Cloth Simulation Function (Zhang et al 2016). CSF uses an algorithm which turns the point-cloud upside down and lets a simulated cloth cling onto the reversed surface. Points in the closest proximity of this draped cloth are then classified as ground points. When the classified ground points were loaded into our Julia code, we found the GroundZ value with the X and Y coordinates of TopVox. To prevent errors, we calculated the GroundZ as an absolute value of the mean of the ground Z coordinates within a two-by-two-meter square buffer.

$$G - T_{height} = TopVox_Z + \left|\overline{Ground_Z}\right| \quad (3)$$

The whole process of measuring the G-T height was wrapped inside of the gt_height() function which performs the calculation itself using the mentioned functions – see Appendix 1. Both the raytracing analysis and the follow-up G-T height calculation can be traced in the flowchart diagram (Figure 7) inserted below.

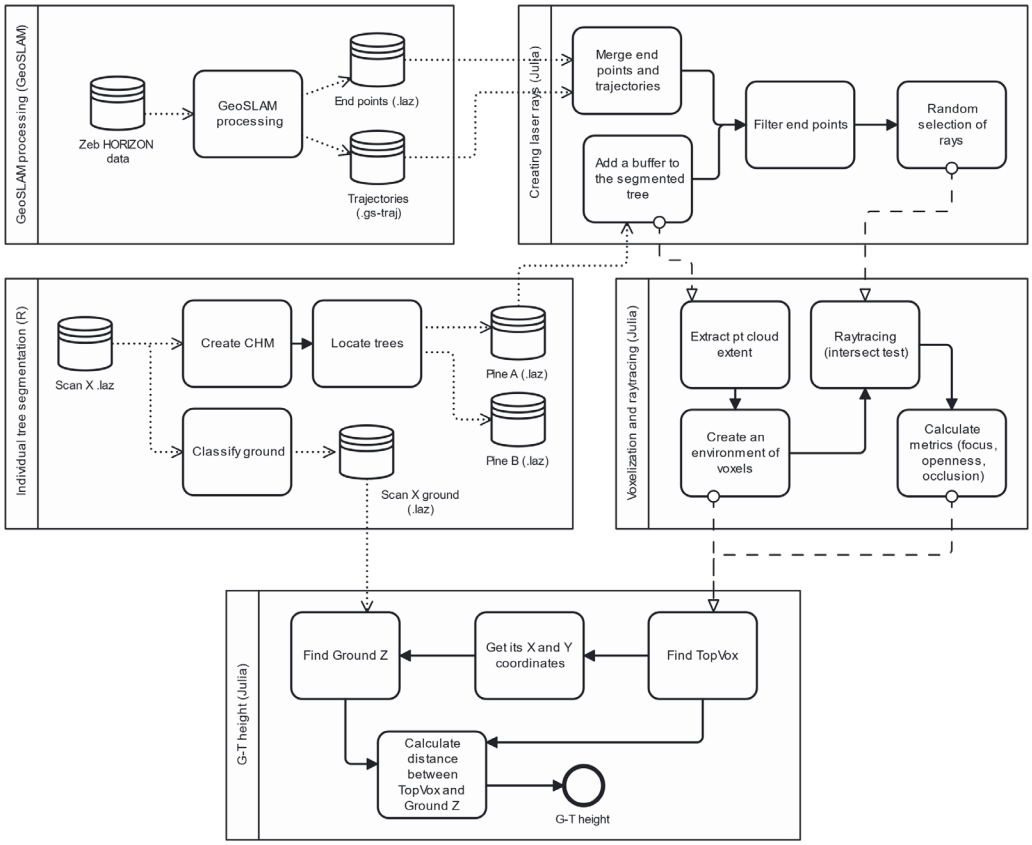



*Figure 7. Flow chart diagram of the raytracing analysis with the G-T height estimation.*

# 3. Results

DISCLAIMER – The raytracing package with the analysis code can be found on our GitHub page: https://github.com/JanZrn/ForesTRACE.jl.

## 3.1 Voxelization

We performed a successful voxelization and ray tracing analysis for tree different voxel resolutions, 10 cm, 25 cm, and 50 cm voxel side lengths. Voxel resolution played a significant role in the time needed to perform the analysis. Fastest analysis was performed for the 50 cm voxels with around 2 hours to analyse 400,000 rays. Voxels with 25 cm sides took around 20 to 25 hours to compute and the highest voxel resolution took around 100 hours, again with 400,000 randomly selected rays. The analysis was performed for 5 scans at a time on a regular workstation in the university computer lab. From Figure 8, we can see the change in voxelization quality for two different scans of the same tree.

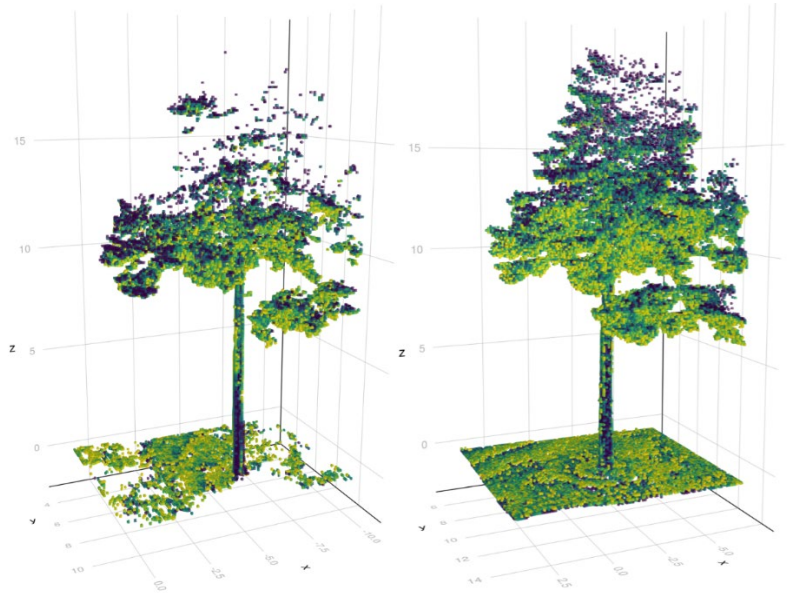


*Figure 8. Side by side comparison of scan quality and voxel openness of pine B. Scan on the left shows scan "static" and on the right "tops". The voxel side length is 10 cm. The lighter the colour (from dark purple to yellow), the higher the voxels openness is and vice versa.*

## 3.2 User bias – Focus

In the Figure 9, we can see that the simulated scanner operator errors/biases are clearly visible with the Focus metric we calculated using the results of our raytracing analysis. On both trees we can see a descending pattern of user focus. Scans in which we intentionally simulated operator errors show higher focus standard deviation than those that were supposed to be error-free. Focus standard deviation on the Y axis represents the mean results of the analysis for 10, 25 and 50 cm voxel size. For both pines the scan "bottoms", where the operator walked close to the trees and intentionally scanned bottoms of the trees, shows the highest user bias. Scans "static" and "line" show the second highest mean focus variability, for Pine A the scan "line" resulted in a higher mean standard deviation and for Pine B scan "static" has higher variability. Scans "control" and "tops show the least amount of variability in user focus, meaning the distribution of voxels within the scanned area is presumed to be more uniform than the beforementioned scans.

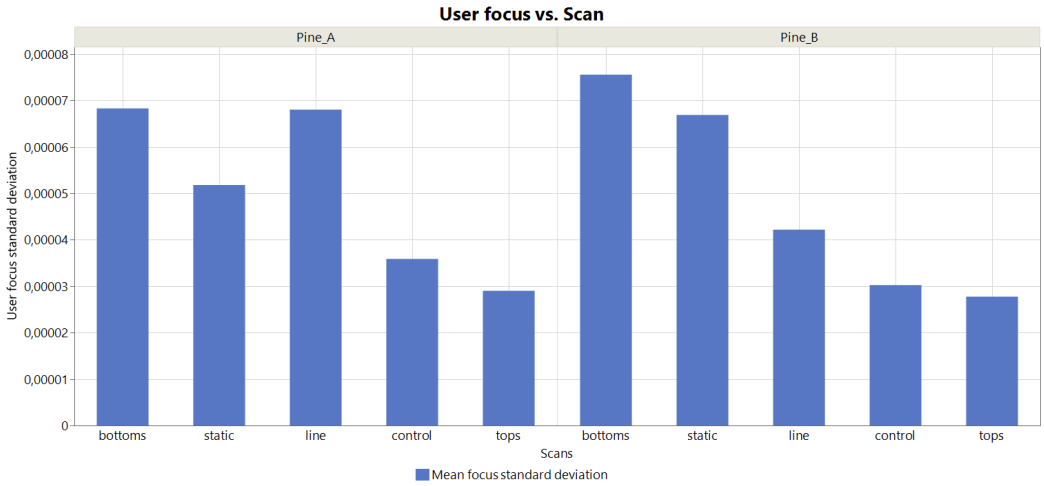



*Figure 9. Histogram of the mean standard deviation of the Focus metric. Mean is taken from all three voxel resolutions for each scan.*

## 3.3 Occlusion

Thanks to the results of the raytracing analysis, we could compute individual occlusion rates of the scans as the percentage of not scanned voxels from the entire voxel environment. The finest voxel resolution (10 cm voxel side length) resulted in the highest the highest occlusion rates in all but one scan – the control scan, where no intentional operator errors were included (See Figure 10). The "static" scan produced the highest percentage of unscanned area, followed by "line" and "bottoms." The "line" scan presented an anomaly in form of almost identical occlusion rate for 25 and 50 cm voxel resolution. On the other hand, in the "control"

scan the 25 cm voxel analysis delivered lower occlusion rate than that of the 10 cm voxels.

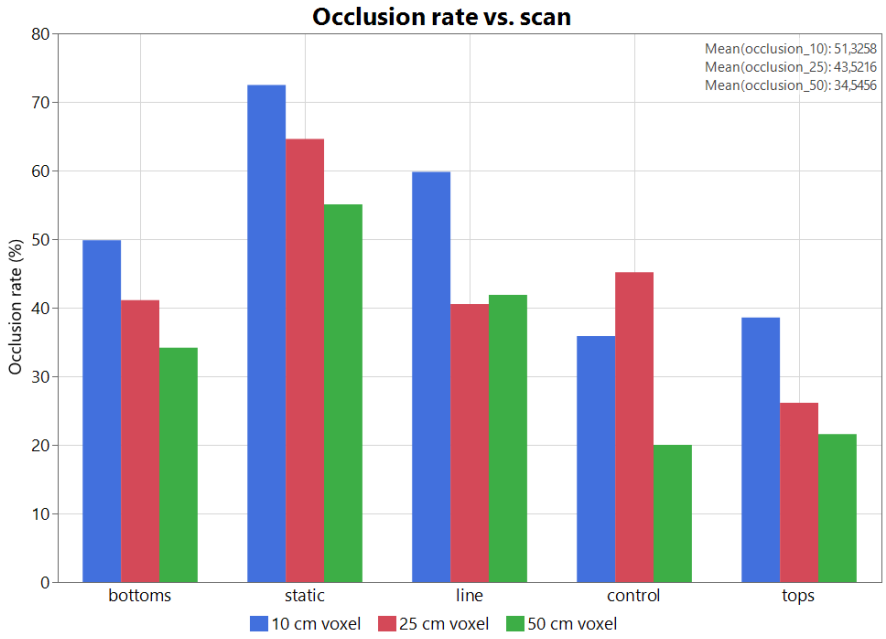



*Figure 10. Bar chart of the occlusion rate in the performed scans dependant on the voxel resolution. Caption box in the top right corner informs about the mean occlusion rate for each voxel resolution. The scans are ordered on the X axis in a descending order of focus variability.*

## 3.4 Tree height

In the Figure 11, we present tree height calculated using four methods of acquisition. Methods *Zmax*, *P95* an *P90* represent metrics derived from pure point cloud analysis using the R *lidRmetrics* package and G-T is our raytracing analysis method. Mean tree height measured with a traditional method using a hypsometer is added as a reference line. Not only do the two quantile methods ($90^{th}$ and $95^{th}$ percentile) substantially underestimate the real height if compared with the traditional hypsometer method, but Figure 11 also show how susceptible they are to user focus, as the estimated height is reported higher with descending user focus. *Zmax* and *G-T* method heights are in proximity of results using a hypsometer. The method of tree height calculation using the raytracing analysis (*G-T method*) seems to be on par with the *Zmax* method, which takes the highest point of the point cloud, but shows slightly higher variability in its results as can be seen in the caption box in the bottom right corner. The standard deviation of *Zmax* is 0.43 and of G-T 0.51. The 10 cm voxel resolution seems to be the least affected by user focus, as its standard deviation is the lowest and it also predicts the tree height to be closest to the height measured in situ (See Figure 12). Lower resolutions show higher variability of the results for the 5 scans and underestimate the height more, if compared to the 10 cm voxel results. Mean height for all three voxel resolutions is

as follows: For 50 cm voxels 18.2 m, for 25 cm voxels 18.4 m and for 10 cm 18.6 m, the mean tree height of the reference hypsometer measurement is 19.1 m. Mean height of the point cloud analysis without ground points resulted in: Zmax 18.7 m, 95$^{th}$ percentile 12.8 m and for 90$^{th}$ percentile 11.7 m.

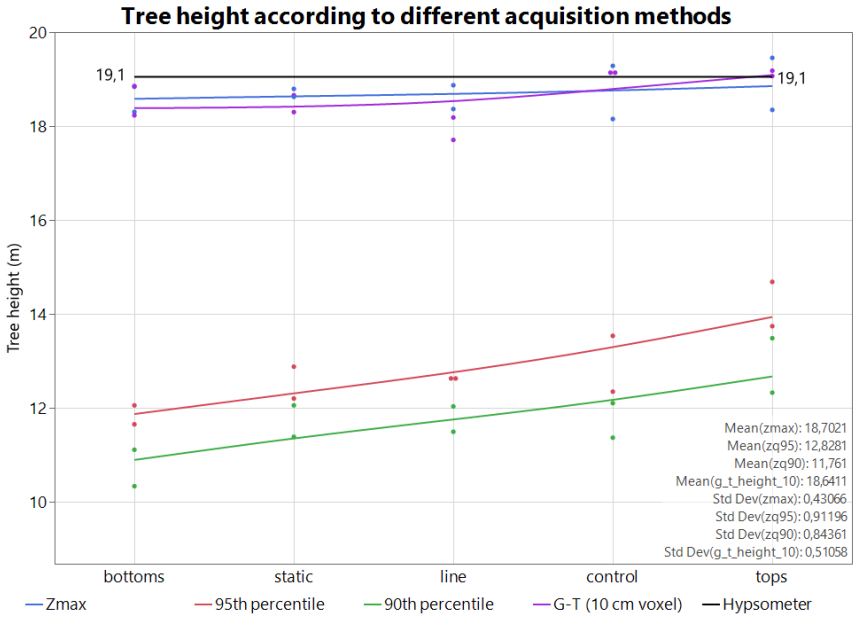



*Figure 11. Tree height according to different acquisition methods. Comparison of Zmax, P95, P90, G-T (10 cm voxel resolution) and a hypsometer height as a reference. Caption in the bottom right corner shows the mean height and the variability of the results using the given methods. The scans are ordered on the X axis in a descending order of focus variability.*

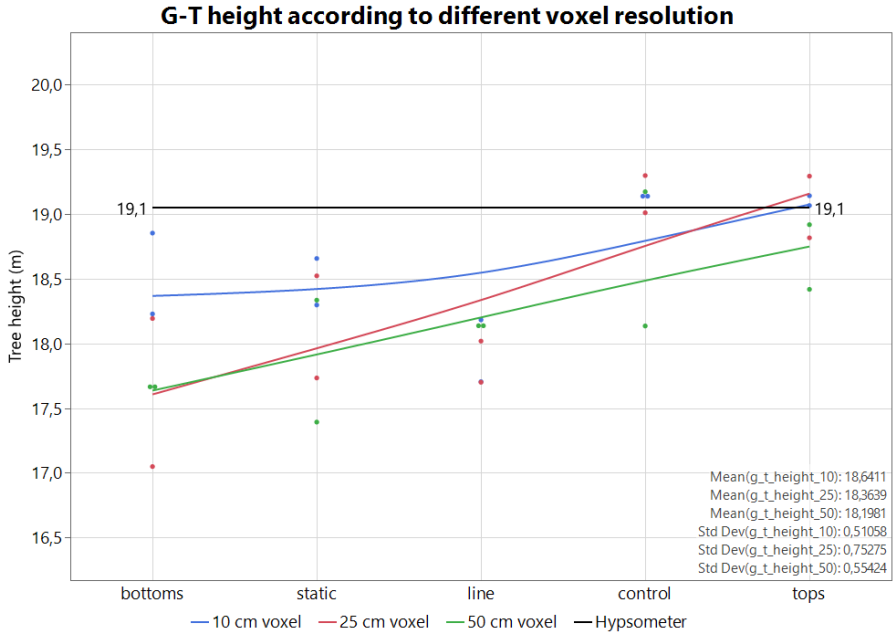



*Figure 12. G-T height according to different voxel resolution. Comparison of 10 cm, 25 cm and 50 cm voxel resolution with the height measured with a hypsometer as a reference. The caption box in the bottom right corner shows the mean height, as well as variability of the results.*

# 4. Discussion

In this thesis, we present a new idea for quantifying LiDAR point cloud occlusion, and other metrics, through raytracing analysis. The goal of this project was to create an open-source raytracing package within the programming language Julia, for the remote sensing community. Conventional methods of LiDAR point cloud analysis take into consideration only the end points of the laser beams, deriving the user defined metrics according to them. Herein presented method voxelizes the point-clouds similarly to established method. However, unlike them, it takes into consideration the trajectory of the laser beam and the space it has passed through. Thanks to this information, we can detect occluded voxels and determine the openness of a given voxel. The mentioned conventional methods cannot accurately predict either of these quantifiers.

In contrast to other voxel-based LS methods, our method is also able to quantify volumetric information about the open space in the plot. In a normal LS point cloud, open (negative) spaces can be identified, but true open space and occluded space are indistinguishable, as we have previously stated – occlusion is one of the major drawbacks while using RS methods. Although the raytracing analysis was performed with H-MLS, we firmly believe our method can be broaden to other LS data acquiring methods, such as drone-based UAV-LS.

There is an ongoing discussion within the remote sensing community about what is an "optimal" scan and how to quantify the point cloud quality (Calders et al., 2020). So far, no consensus has been reached and therefore we present the user focus quantifier as it could be the deciding quantifier for what an optimal data set could be. With the user focus quantifier, we can analyse if the area of interest is scanned homogenously or if the operator missed certain parts and fixated on other areas of the plot. Ideally, the plot should be scanned as consistently as possible, to reduce the occlusion effect and increase the scanning completeness of the trees (Bauwens et al., 2016). With the ray tracing approach, we were able to demonstrate the effect of human error introduced by not choosing the adequate walking path with the scanner or if the operator focused on one part of the scanned area more than the rest. We discuss that the introduced focus quantifier is of a similar nature to the off-nadir scanning angle while performing ALS. It was proven, that higher scanning angles while airborne scanning affect the results. The research shows that higher

scanning angle results in higher underestimation of results, namely tree height and gap fractious, as well as other structure metrics (Holmgren et al., 2003; Liu et al., 2018). The operator focus variability we quantified with the raytracing approach affected both the occlusion rate and the resulting tree height. However, the occlusion rate in our case is tied to the chosen voxel resolution as well. Using this Focus quantifier, determining an optimal walking path for various environments using a H-PLS can be the subject of further studies. This has been done to extent done with the TLS scanners (L. Li et al., 2021) and ALS (Næsset, 2009), but there is a lack of research for the H-PLS platform.

García et al. (2011) discusses that the voxel-based approaches are a common tool for volumetric point cloud calculations, but they also state that voxel resolution affects the result. However, relatively few studies on the effect of voxel size were carried out (Ross et al., 2022). We have tested our raytracing method on three different voxel resolutions – 10-, 25- and 50-centimetre voxel side length. And both the occlusion rate and the tree height were impacted by the resolution. The tree height acquired using the G-T method was only slightly affected by the resolution, as will be discussed later. However, the occlusion rate is strongly influenced by the voxel size. Higher voxel resolution (smaller voxels) demonstrated higher occlusion rates for the same scan in comparison to bigger voxels. The mean occlusion for 50 cm voxels was 34.6 %, for 25 cm 43.5 % and for 10 cm voxels 51.3 %. This can be explained by the voxels with lower resolution having bigger size and thus reaching into space that would be qualified as occluded with higher voxel resolution. This result agrees with Mathes et al. (2023), where they stated the optimal voxel resolution for minimizing occlusion while still providing good results detail is to be around 20 cm or less. Herein, we demonstrated that voxel resolution effects the occlusion, as also observed by Mathes et al. (2023) using the same H-PLS scanning approach. Their conclusion is that higher voxel size mitigates occlusion but lowers the calculated metric precision. The G-T height with changing voxel size agrees with their tested hypothesis, as visible from Figure 12.

However, according to García et al. (2011) if voxels are too small, the void between the points may be misinterpreted, potentially skewing the results if not appropriately filled. The voids in the voxelized point cloud can be interpreted twofold, either because the object within it was occluded or due to a lack of objects. Thanks to the ray tracing approach, one would be able to distinguish the nature of the void – whether it was occluded from view or actually free of objects, thereby providing more insight about the environment. The understanding of occlusion, according to García et al. (2011), is also useful for further filling in these empty voxels with artificial information based on their surroundings. They describe this case for the Plant Area Density (PAD) measures with the AMAPVox tool (Vincent et al., 2017), in which there is an option to fill occluded voxels for ALS data but

has not yet been implemented for ground-based scans. Voxelized forest environment, with filled in information in the occluded voxels can also further help the research community working on Radiative Transfer Models (RTMs), which are being developed to better understand the processes driving the biochemistry of forest stands. The methods of forest ecosystem management should take these environmental drivers into consideration to actively help with climate change mitigation, as forests affect microclimate within them and help buffer temperature cycles in their vicinity (Aussenac, 2000; Ligot et al., 2014). RTMs link forest structure, ranging from 1D to 3D, with the light absorption and reflection mechanisms. Data needed to describe the forest structure for RTMs are pieced together from field inventorying and ALS data. However, some gaps in the parameters based on traditional field inventories are theorized to be filled with some ground-based LS approach (Calders et al., 2020). These gaps may include, given as examples by Calders et al. (2020), leaf distribution and density within the canopy, crown shapes or branching angles of the habitus (wooden skeleton). If these metrics could be calculated using voxel openness, as introduced in this thesis, is a question for further research but we believe there is a promising potential. The openness quantifier describes the density of points within the voxel and number of rays passing through it – hence models to predict for example the beforementioned leaf distribution inside of the canopy can be created in the future efforts.

As discussed earlier, the absence of return points in a part of the point cloud is explained by either the object in this space being occluded from the view of the scanner or by the lack of objects the laser could return from. Due to the chosen test method in our project, a voxel was deemed occluded if no rays passed through it. As such, one problem we came across during our analysis was that we could not tell apart voxels that were occluded from the view of the scanner and those being intersected by rays that left the scanned area without returning. This issue can be explained by the GeoSLAM product. If certain endpoints are absent, due to the rays leaving without returning, the trajectory cannot be paired with anything and is consequently ignored in the merging process. A significant fraction of all occluded voxels were located in the higher parts of the voxel environment, as the laser beams could not return from the object-free sky. The high occlusion rates are, therefore, specific to the demonstration site we selected. The demonstrational site was chosen so that the operator had free range of movement and clear view of the crown tops, hence the objects in the vicinity of the trees were sparse and the laser beams, apart from those returning from the two pines and the ground, had nowhere to return from. To mitigate this issue into some extent, a 5-meter buffer was added to extract end points from the surrounding of the segmented trees. We can presume, that this effect would be lower in an actual forest stand, where the density of objects is significantly higher than in that of a city park setting. Size of the “miss mitigation” buffer could be an interest of further studies on how it affects the runtime of the

analysis, as well as its results. This problem is mainly an issue of ground-based scanners, as we presume that ALS and UAV-LS point clouds would not be affected as much. ALS and UAV-LS scans in a downward motion to scan the canopy/ground, however ground-based scanners scan the other way around and thus have a high potential of rays leaving and never returning. Another potential issue could be our decision to randomly select 400,000 rays for analysis. This decision was made to save time during the test. If we tested all rays, as done by Schneider et al. (2019), the analysis may have yielded different results. We are aware that the runtime of the raytracing analysis is still quite high, if we compare it to Schneider et al. (2019), where the team ran the analysis in C++ (code unknown) and was able to test more than 90 million beams on a 60x60 m plot with voxel resolution of 10x10 cm in around three days. Our code has not yet reached this speed level and requires further optimization, namely manually allocating mutable memory and making multithreading viable. Their analysis also combined point clouds from both TLS and ALS methods, thus had a better coverage of the crowns – something our analysis lacked.

Although our analysis suffered from limited coverage of the crowns, resulting in a significant underestimation of tree height using the quantile methods, the G-T method we introduce demonstrated potential for future applications. In all 5 scans for both trees the G-T method was affected by the operator bias only to some extent, as well as showed relatively low variability and result similarity, if compared to the traditional method using a hypsometer, unlike the percentile methods. The results of the quantile approach align with previous studies indicating that ground-based scanning generally results in height underestimation (García et al., 2011). This phenomenon is primarily attributed to the occlusion of the tops by the lower branches of the canopy, which also results in a higher point density, thereby placing greater emphasis on the lower part of the canopy with the quantile approach. The quantile approach's resulting underestimation is noticeable, even though the selected demonstrational site lacked understory hedges or tall grasses. This effect could be even more pronounced if the scanning took place in a forest plot with dense understory and was not addressed during post-processing. Unlike the quantile methods, the introduced raytracing G-T approach does not face this problem, eliminating the need to filter points from the lower parts of the point cloud, even in the presence of a dense understory. The introduced G-T height calculation still resulted in a slight height underestimation, if compared to the hypsometer approach, but much closer to the presumed height. Figure 11 also highlights the non-negligible spread of results for both of the height quantile methods with changing user focus variability. The standard deviation of the calculated height for $95^{th}$ and $90^{th}$ percentile was 0.91 and 0.84 respectively. This result variability can be discussed as susceptibility of derived height to quality of a given point cloud, defined by the *Focus* quantifier. Even though the quantile methods were introduces

as a mean to describe biomass on a stand level, they are also used in the RS community to measure top height of stands (Mao et al., 2019). We can presume, that this susceptibility of quantile methods for measuring top height to human error and point cloud quality on individual tree level, as described in this project, will be noticeable even on the stand level. However, it has not been tested and needs further research. The variability of the G-T height resulted in a slightly higher variability, if compared to the Zmax method (0.43 for Zmax and 0.51 for G-T with 10 cm voxels), but the variability response to user focus resulted in about a half of the quantile approach. This slightly higher variability of G-T, compared to Zmax, can be explained by a failsafe included in the G-T method. This failsafe allows a voxel to be the TopVox one only if there are at least two returns from within the voxel. It should negate scanner errors and outliers occasionally present in point clouds, something the Zmax method lacks. Another advantage of the G-T method is that it does not require point cloud height normalization, unlike some established methods. It is advised against using point cloud height normalization as it affects the retrieved plant metrics in sloped stands as demonstrated on PAD by Liu et al. (2017). The mentioned topographical point cloud normalization moves the points from their original position and thus the relationships between points, based on their position in space, cannot be interpreted. These relationships between points (or voxels) could indicate branching angles, canopy shape or stem growth curve. And with the orientation and distance links severed, the derived results are presumably incorrect, hence non-normalizing methods should be favoured by the RS community.

The resulting G-T height variability for 50 cm voxels (0.55) corresponded to that of the 10 cm voxels (0.51), but it was higher for the 25 cm voxels (0.75). We explain this result, and perhaps even the resulting variability of the other resolutions, as the effect of the random ray selection. We chose to analyse random 400,000 rays for time sakes, but if the raytracing is further optimized and we can analyse more rays, perhaps all, we presume higher result precision will be achieved. We presume even better results would be achieved, if the ground-based data was combined with above canopy data, like TLS on top of a crane as demonstrated by Schneider et al. (2019) or UAV-LS. UAV-LS data, with its high ray density in the crown tops, could better describe the voxel openness quantifier, as explained in the preceding paragraphs. As stated in the introduction, the flight altitude for UAV-LS and ALS is different and results in differing spatial resolution. Therefore, we argue against using raytracing methods with the ALS platform. The ALS flight altitude is typically up to several kilometres and thus its laser beam is considered a cone and not a line, which is the premise of the raytracing approach. However, small-footprint drone-based laser scanning methods are arguably good candidates for further raytracing trials using the introduced package.

We firmly believe that the height method we present may bear flaws, but it should inspire members of the remote sensing and forest practice communities to come up with new ideas how to use raytracing results and introduced quantifiers. The package we present, and plan to continue adjusting and optimizing, should stand as an easy-to-use base for researchers and forest management practice. The vision for the tool is to be further developed and implemented not only by the forest RS community, as the raytracing approach appears to be the next step in quantifying the environment around us. We believe that the approach presented by the package can be adopted by the rest of LS acquisition platforms and methods, not only by H-PLS as used in our demonstration. However, the footprint size and beam divergence should be considered. A potential use for the raytracing approach and ForesTRACE.jl package introduced in this project could be quality control of point clouds while inventorying. The longer analysis time would make larger number of point clouds to be raytraced ineffective, however quality control on a smaller sample would be viable. With this approach, the homogeneity of the scanned inventory could be verified. Another example of use for the ForesTRACE.jl package is exploring the ways of quantifying the negative space within the scanned area. Open space quantification is not possible with the current method of simply analysing the end points in a point clouds. Alternatively, it could help with species habitus recognition as its voxels can represent the tree structure, with added value in the voxel openness value which could represent either woody or leafy parts of the trees. These ideas were not developed further in this thesis, as it would the time limit for this project.

# References


Abegg, M., Kükenbrink, D., Zell, J., Schaepman, M. E., & Morsdorf, F. (2017). Terrestrial laser scanning for forest inventories-tree diameter distribution and scanner location impact on occlusion. Forests, 8(6). https://doi.org/10.3390/f8060184

Almeida, D. R. A. de, Stark, S. C., Shao, G., Schietti, J., Nelson, B. W., Silva, C. A., Gorgens, E. B., Valbuena, R., Papa, D. de A., & Brancalion, P. H. S. (2019). Optimizing the Remote Detection of Tropical Rainforest Structure with Airborne Lidar: Leaf Area Profile Sensitivity to Pulse Density and Spatial Sampling. Remote Sensing, 11(1), 92. https://doi.org/10.3390/rs11010092

Andersen, H.-E., Reutebuch, S. E., & McGaughey, R. J. (2006). A rigorous assessment of tree height measurements obtained using airborne lidar and conventional field methods. Canadian Journal of Remote Sensing, 32(5), 355–366. https://doi.org/10.5589/m06-030

Aussenac, G. (2000). Interactions between forest stands and microclimate: Ecophysiological aspects and consequences for silviculture. Annals of Forest Science, 57(3), 287–301. https://doi.org/10.1051/FOREST:2000119

Balenović, I., Liang, X., Jurjević, L., Hyyppä, J., Seletković, A., & Kukko, A. (2020). Hand-held personal laser scanning – current status and perspectives for forest inventory application. Croatian Journal of Forest Engineering, 42(1), 165–183. https://doi.org/10.5552/crojfe.2021.858

Bauwens, S., Bartholomeus, H., Calders, K., & Lejeune, P. (2016). Forest inventory with terrestrial LiDAR: A comparison of static and hand-held mobile laser scanning. Forests, 7(6). https://doi.org/10.3390/f7060127

Béland, M., Widlowski, J.-L., Fournier, R. A., Côté, J.-F., & Verstraete, M. M. (2011). Estimating leaf area distribution in savanna trees from terrestrial LiDAR measurements. Agricultural and Forest Meteorology, 151(9), 1252–1266. https://doi.org/10.1016/j.agrformet.2011.05.004

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A Fresh Approach to Numerical Computing. SIAM Review, 59(1), 65–98. https://doi.org/10.1137/141000671

Bienert, A., Hess, C., Maas, H.-G., & von Oheimb, G. (2014). A voxel-based technique to estimate the volume of trees from terrestrial laser scanner data. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XL–5, 101–106. https://doi.org/10.5194/isprsarchives-XL-5-101-2014

Bittner, S., Gayler, S., Biernath, C., Winkler, J. B., Seifert, S., Pretzsch, H., & Priesack, E. (2012). Evaluation of a ray-tracing canopy light model based on terrestrial laser scans. Canadian Journal of Remote Sensing, 38(5), 619–628. https://doi.org/10.5589/m12-050

Bouchet-Valat, M., & Kamiński, B. (2023). DataFrames.jl: Flexible and Fast Tabular Data in Julia. Journal of Statistical Software, 107(4). https://doi.org/10.18637/jss.v107.i04

Calders, K., Adams, J., Armston, J., Bartholomeus, H., Bauwens, S., Bentley, L. P., Chave, J., Danson, F. M., Demol, M., Disney, M., Gaulton, R., Krishna Moorthy, S. M., Levick, S. R., Saarinen, N., Schaaf, C., Stovall, A., Terryn, L., Wilkes, P., & Verbeeck, H. (2020). Terrestrial laser scanning in forest ecology: Expanding the horizon. In Remote Sensing of Environment (Vol. 251). Elsevier Inc. https://doi.org/10.1016/j.rse.2020.112102

Calders, K., Newnham, G., Burt, A., Murphy, S., Raumonen, P., Herold, M., Culvenor, D., Avitabile, V., Disney, M., Armston, J., & Kaasalainen, M. (2015). Nondestructive estimates of above-ground biomass using terrestrial laser scanning. Methods in Ecology and Evolution, 6(2), 198–208. https://doi.org/10.1111/2041-210X.12301

Danisch, S., & Krumbiegel, J. (2021). Makie.jl: Flexible high-performance data visualization for Julia. Journal of Open Source Software, 6(65), 3349. https://doi.org/10.21105/joss.03349

De Frenne, P., Lenoir, J., Luoto, M., Scheffers, B. R., Zellweger, F., Aalto, J., Ashcroft, M. B., Christiansen, D. M., Decocq, G., De Pauw, K., Govaert, S., Greiser, C., Gril, E., Hampe, A., Jucker, T., Klinges, D. H., Koelemeijer, I. A., Lembrechts, J. J., Marrec, R., … Hylander, K. (2021). Forest microclimates and climate change: Importance, drivers and future research agenda. Global Change Biology, 27(11), 2279–2297. https://doi.org/10.1111/GCB.15569

Fassnacht, F. E., White, J. C., Wulder, M. A., & Næsset, E. (2023). Remote sensing in forestry: current challenges, considerations and directions. Forestry: An International Journal of Forest Research. https://doi.org/10.1093/forestry/cpad024

García, M., Danson, F. M., Riaño, D., Chuvieco, E., Ramirez, F. A., & Bandugula, V. (2011). Terrestrial laser scanning to estimate plot-level forest canopy fuel properties. International Journal of Applied Earth Observation and Geoinformation, 13(4), 636–645. https://doi.org/10.1016/j.jag.2011.03.006

Goodwin, N. R., Coops, N. C., & Culvenor, D. S. (2006). Assessment of forest structure with airborne LiDAR and the effects of platform altitude. Remote Sensing of Environment, 103(2), 140–152. https://doi.org/10.1016/j.rse.2006.03.003

Hanberry, B. B., Palik, B. J., & He, H. S. (2012). Comparison of historical and current forest surveys for detection of homogenization and mesophication of Minnesota forests. Landscape Ecology, 27(10), 1495–1512. https://doi.org/10.1007/s10980-012-9805-5

Holmgren, J., Nilsson, M., & Olsson, H. (2003). Simulating the effects of lidar scanning angle for estimation of mean tree height and canopy closure. Canadian Journal of Remote Sensing, 29(5), 623–632. https://doi.org/10.5589/m03-030

Hyyppä, E., Yu, X., Kaartinen, H., Hakala, T., Kukko, A., Vastaranta, M., & Hyyppä, J. (2020). Comparison of Backpack, Handheld, Under-Canopy UAV, and Above-Canopy UAV Laser Scanning for Field Reference Data Collection in Boreal Forests. Remote Sensing, 12(20), 3327. https://doi.org/10.3390/rs12203327

Hyyppä, J., Hyyppä, H., Yu, X., Kaartinen, H., Kukko, A., & Holopainen, M. (2008). Forest Inventory Using Small-Footprint Airborne LiDAR. In Topographic Laser Ranging and Scanning (pp. 335–370). CRC Press. https://doi.org/10.1201/9781420051438.ch12

Jurjević, L., Liang, X., Gašparović, M., & Balenović, I. (2020). Is field-measured tree height as reliable as believed – Part II, A comparison study of tree height estimates from conventional field measurement and low-cost close-range remote sensing in a deciduous forest. ISPRS Journal of Photogrammetry and Remote Sensing, 169, 227–241. https://doi.org/10.1016/j.isprsjprs.2020.09.014

Larrieu, L., Paillet, Y., Winter, S., Bütler, R., Kraus, D., Krumm, F., Lachat, T., Michel, A. K., Regnery, B., & Vandekerkhove, K. (2018). Tree related microhabitats in temperate and Mediterranean European forests: A hierarchical typology for inventory standardization. Ecological Indicators, 84, 194–207. https://doi.org/10.1016/j.ecolind.2017.08.051

Li, L., Mu, X., Soma, M., Wan, P., Qi, J., Hu, R., Zhang, W., Tong, Y., & Yan, G. (2021). An Iterative-Mode Scan Design of Terrestrial Laser Scanning in Forests for Minimizing Occlusion Effects. IEEE Transactions on Geoscience and Remote Sensing, 59(4), 3547–3566. https://doi.org/10.1109/TGRS.2020.3018643

Li, W., Guo, Q., Jakubowski, M. K., & Kelly, M. (2012). A New Method for Segmenting Individual Trees from the Lidar Point Cloud. Photogrammetric Engineering & Remote Sensing, 78(1), 75–84. https://doi.org/10.14358/PERS.78.1.75

Liang, X., Kukko, A., Kaartinen, H., Hyyppä, J., Yu, X., Jaakkola, A., & Wang, Y. (2014). Possibilities of a Personal Laser Scanning System for Forest Mapping and Ecosystem Services. Sensors, 14(1), 1228–1248. https://doi.org/10.3390/s140101228

Ligot, G., Balandier, P., Courbaud, B., & Claessens, H. (2014). Forest radiative transfer models: which approach for which application? Canadian Journal of Forest Research, 44(5), 391–403. https://doi.org/10.1139/cjfr-2013-0494

Liu, J., Skidmore, A. K., Heurich, M., & Wang, T. (2017). Significant effect of topographic normalization of airborne LiDAR data on the retrieval of plant area index profile in mountainous forests. ISPRS Journal of Photogrammetry and Remote Sensing, 132, 77–87. https://doi.org/10.1016/j.isprsjprs.2017.08.005

Liu, J., Skidmore, A. K., Jones, S., Wang, T., Heurich, M., Zhu, X., & Shi, Y. (2018). Large off-nadir scan angle of airborne LiDAR can severely affect the estimates of forest structure metrics. ISPRS Journal of Photogrammetry and Remote Sensing, 136, 13–25. https://doi.org/10.1016/j.isprsjprs.2017.12.004

Luo, Y., Xie, D., Qi, J., Zhou, K., Yan, G., & Mu, X. (2023). LESS LiDAR: A Full-Waveform and Discrete-Return Multispectral LiDAR Simulator Based on Ray Tracing Algorithm. Remote Sensing, 15(18), 4529. https://doi.org/10.3390/rs15184529

Mallet, C., & Bretar, F. (2009). Full-waveform topographic lidar: State-of-the-art. ISPRS Journal of Photogrammetry and Remote Sensing, 64(1), 1–16. https://doi.org/10.1016/j.isprsjprs.2008.09.007

Mao, L., Bater, C. W., Stadt, J. J., White, B., Tompalski, P., Coops, N. C., & Nielsen, S. E. (2019). Environmental landscape determinants of maximum forest canopy height of boreal forests. Journal of Plant Ecology, 12(1), 96–102. https://doi.org/10.1093/jpe/rtx071

Mathes, T., Seidel, D., Häberle, K.-H., Pretzsch, H., & Annighöfer, P. (2023). What Are We Missing? Occlusion in Laser Scanning Point Clouds and Its Impact on the Detection of Single-Tree Morphologies and Stand Structural Variables. Remote Sensing, 15(2), 450. https://doi.org/10.3390/rs15020450

Möller, T., & Trumbore, B. (1997). Fast, Minimum Storage Ray-Triangle Intersection. Journal of Graphics Tools, 2(1), 21–28. https://doi.org/10.1080/10867651.1997.10487468

Morsdorf, F., Frey, O., Koetz, B., & Meier, E. (2007). RAY TRACING FOR MODELING OF SMALL FOOTPRINT AIRBORNE LASER SCANNING RETURNS. https://doi.org/https://doi.org/10.3929/ethz-b-000107380

Næsset, E. (2009). Effects of different sensors, flying altitudes, and pulse repetition frequencies on forest canopy metrics and biophysical stand properties derived from small-footprint airborne laser data. Remote Sensing of Environment, 113(1), 148–159. https://doi.org/10.1016/j.rse.2008.09.001

Næsset, E., & Bjerknes, K.-O. (2001). Estimating tree heights and number of stems in young forest stands using airborne laser scanner data. Remote Sensing of Environment, 78(3), 328–340. https://doi.org/10.1016/S0034-4257(01)00228-0

Pimont, F., Allard, D., Soma, M., & Dupuy, J. L. (2018). Estimators and confidence intervals for plant area density at voxel scale with T-LiDAR. Remote Sensing of Environment, 215, 343–370. https://doi.org/10.1016/j.rse.2018.06.024

Puliti, S., Ørka, H., Gobakken, T., & Næsset, E. (2015). Inventory of Small Forest Areas Using an Unmanned Aerial System. Remote Sensing, 7(8), 9632–9654. https://doi.org/10.3390/rs70809632

Ross, C. W., Loudermilk, E. L., Skowronski, N., Pokswinski, S., Hiers, J. K., & O'Brien, J. (2022). LiDAR Voxel-Size Optimization for Canopy Gap Estimation. Remote Sensing, 14(5), 1054. https://doi.org/10.3390/rs14051054

Roussel, J.-R., Auty, D., Coops, N. C., Tompalski, P., Goodbody, T. R. H., Meador, A. S., Bourdon, J.-F., de Boissieu, F., & Achim, A. (2020). lidR: An R package for analysis of Airborne Laser Scanning (ALS) data. Remote Sensing of Environment, 251, 112061. https://doi.org/10.1016/j.rse.2020.112061

Saarinen, N., Kankare, V., Vastaranta, M., Luoma, V., Pyörälä, J., Tanhuanpää, T., Liang, X., Kaartinen, H., Kukko, A., Jaakkola, A., Yu, X., Holopainen, M., & Hyyppä, J. (2017). Feasibility of Terrestrial laser scanning for collecting stem

volume information from single trees. ISPRS Journal of Photogrammetry and Remote Sensing, 123, 140–158. https://doi.org/10.1016/j.isprsjprs.2016.11.012

Schneider, F. D., Kükenbrink, D., Schaepman, M. E., Schimel, D. S., & Morsdorf, F. (2019). Quantifying 3D structure and occlusion in dense tropical and temperate forests using close-range LiDAR. Agricultural and Forest Meteorology, 268, 249–257. https://doi.org/10.1016/j.agrformet.2019.01.033

Sferlazza, S., Maltese, A., Dardanelli, G., & La Mela Veca, D. S. (2022). Optimizing the Sampling Area across an Old-Growth Forest via UAV-Borne Laser Scanning, GNSS, and Radial Surveying. ISPRS International Journal of Geo-Information, 11(3), 168. https://doi.org/10.3390/ijgi11030168

Solberg, S., & Strand, L. (1999). Crown density assessments, control surveys and reproducibility. Environmental Monitoring and Assessment, 56(1), 75–86. https://doi.org/10.1023/A:1005980326079

Stark, S. C., Leitold, V., Wu, J. L., Hunter, M. O., de Castilho, C. V., Costa, F. R. C., McMahon, S. M., Parker, G. G., Shimabukuro, M. T., Lefsky, M. A., Keller, M., Alves, L. F., Schietti, J., Shimabukuro, Y. E., Brandão, D. O., Woodcock, T. K., Higuchi, N., de Camargo, P. B., de Oliveira, R. C., & Saleska, S. R. (2012). Amazon forest carbon dynamics predicted by profiles of canopy leaf area and light environment. Ecology Letters, 15(12), 1406–1414. https://doi.org/10.1111/j.1461-0248.2012.01864.x

Stovall, A. E. L., Vorster, A. G., Anderson, R. S., Evangelista, P. H., & Shugart, H. H. (2017). Non-destructive aboveground biomass estimation of coniferous trees using terrestrial LiDAR. Remote Sensing of Environment, 200, 31–42. https://doi.org/10.1016/j.rse.2017.08.013

Tomppo, E., Gschwantner, T., Lawrence, M., & Mcroberts, R. E. (2010). National Forest Inventories (E. Tomppo, T. Gschwantner, M. Lawrence, & R. E. McRoberts, Eds.). Springer Netherlands. https://doi.org/10.1007/978-90-481-3233-1

Valbuena, R., O'Connor, B., Zellweger, F., Simonson, W., Vihervaara, P., Maltamo, M., Silva, C. A., Almeida, D. R. A., Danks, F., Morsdorf, F., Chirici, G., Lucas, R., Coomes, D. A., & Coops, N. C. (2020). Standardizing Ecosystem Morphological Traits from 3D Information Sources. Trends in Ecology & Evolution, 35(8), 656–667. https://doi.org/10.1016/j.tree.2020.03.006

Vincent, G., Antin, C., Laurans, M., Heurtebize, J., Durrieu, S., Lavalley, C., & Dauzat, J. (2017). Mapping plant area index of tropical evergreen forest by airborne laser scanning. A cross-validation study using LAI2200 optical sensor. Remote Sensing of Environment, 198, 254–266. https://doi.org/10.1016/j.rse.2017.05.034

Von Arx, G., Graf Pannatier, E., Thimonier, A., & Rebetez, M. (2013). Microclimate in forests with varying leaf area index and soil moisture: potential implications for seedling establishment in a changing climate. Journal of Ecology, 101(5), 1201–1213. https://doi.org/10.1111/1365-2745.12121

Wang, Y., Lehtomäki, M., Liang, X., Pyörälä, J., Kukko, A., Jaakkola, A., Liu, J., Feng, Z., Chen, R., & Hyyppä, J. (2019). Is field-measured tree height as reliable as believed – A comparison study of tree height estimates from field measurement, airborne laser scanning and terrestrial laser scanning in a boreal forest. ISPRS

Journal of Photogrammetry and Remote Sensing, 147, 132–145. https://doi.org/10.1016/j.isprsjprs.2018.11.008

White, J. C., Coops, N. C., Wulder, M. A., Vastaranta, M., Hilker, T., & Tompalski, P. (2016). Remote Sensing Technologies for Enhancing Forest Inventories: A Review. In Canadian Journal of Remote Sensing (Vol. 42, Issue 5, pp. 619–641). Taylor and Francis Inc. https://doi.org/10.1080/07038992.2016.1207484

Zellweger, F., Coomes, D., Lenoir, J., Depauw, L., Maes, S. L., Wulf, M., Kirby, K. J., Brunet, J., Kopecký, M., Máliš, F., Schmidt, W., Heinrichs, S., den Ouden, J., Jaroszewicz, B., Buyse, G., Spicher, F., Verheyen, K., & De Frenne, P. (2019). Seasonal drivers of understorey temperature buffering in temperate deciduous forests across Europe. Global Ecology and Biogeography, 28(12), 1774–1786. https://doi.org/10.1111/GEB.12991

Zong, X., Wang, T., Skidmore, A. K., & Heurich, M. (2021). The impact of voxel size, forest type, and understory cover on visibility estimation in forests using terrestrial laser scanning. GIScience & Remote Sensing, 58(3), 323–339. https://doi.org/10.1080/15481603.2021.1873588

# Popular science summary

Documenting the state of the forest property is not an easy task for forest managers. Forest managers usually measure the stand and tree metrics with either direct or indirect methods, using callipers, tape measures and height meters. Unfortunately, some stand metrics quantifying the horizontal and vertical structure cannot be acquired using these physical methods. Thus, remote sensing methods are used, to obtain these otherwise immeasurable metrics and to streamline the work in a more time- and cost-efficient way. One of the widely used remote sensing technologies – laser scanning, is usually classified by the platform it is mounted to, be it in the air or under the canopy. However, all of these platforms suffer from one major drawback – occlusion. Occlusion is a technical term describing an object being hidden from view of the scanner, thus not being registered. This absence of information in the three-dimensional point cloud can skew the results, for example indicating fewer trees in a forest plot or underestimating tree height due to lower branches obstructing the treetops. Another drawback of some LS platforms (mainly the hand-held ones) is the inconsistency among human operators. One operator may focus on one part of the plot while another concentrates on a different area. However, the goal is to achieve scans that are as homogeneous as possible.

ForesTRACE,jl is a newly introduced raytracing package for the programming language Julia, which aims to quantify these errors. Quantifying occlusion and user bias is the first step to scanning optimalization. Unlike traditional point cloud methods, raytracing takes into consideration not only the end points, but merges the endpoints with the scanner's location. The merged laser beams traverse a three-dimensional environment made out of voxels – cubes. After the analysis is performed, we can quantify the cubes which did not interact with any rays, as well as the attention given to each of the cubes, or the openness quantifier (given by the proportion of rays ending in a voxel to rays passing through it). With this information, the forest remote sensing community can decide if a given scan is good enough to work with or needs a re-do, to create tree and stand metrics as close to reality as possible.

In this project, we were able to describe how tree height derived from traditional point cloud analysis is susceptible to user focus variability, unlike the introduced G-T tree height method, which takes advantage of the raytracing quantifiers.

# Acknowledgements

I would like to thank Cameron Pellett, the assistant supervisor of this thesis. He was always happy to help me, share his views and give unending feedback. Thanks to his help I was able to continue coding after a plateau and he came up with ideas to write about that would otherwise not come to my mind. My other gratitude is aimed towards my supervisor Ruben Valbuena and Remote Sensing course leader Jonas Bohlin for making this thesis possible and giving me this great opportunity.

Thank you!

# Appendix 1 – ForesTRACE.jl Documentation

# ForesTRACE.jl

## List of introduced functions


https://github.com/JanZrn/ForesTRACE.jl

Function name: **within()**

| | |
|---|---|
| Arguments: | LasPoint, LasHeader, bounding box (Meshes.Box) |
| Returns: | LasPoint, LasHeader |
| Description: | Identifies what laser points lie within a bounding box (Meshes.Box geometry). The bounding box can be either a 2D or a 3D object. If it is a 2D object, will include all Z values. |

Function **filter_pixel!()**

| | |
|---|---|
| Arguments: | LasPoint, LasHeader, bounding box (Meshes.Box) |
| Returns: | Modified LasPoint, LasHeader |
| Description: | Filters the point cloud to contain only the points within a given bounding box. The bounding box can be either a 2D or a 3D object. If it is a 2D object, will include all Z values. |

Mutable struct **Voxel**

| | |
|---|---|
| Arguments: | .poly, .pass, .stop |
| Returns: | Voxel |
| Description: | A custom mutable struct containing geometry, number of stops and number of passes. It is a cube with a given voxel side length and position in the environment, as determined by the Meshes.Box in .poly. |

Function **create_voxels()**

| | |
|---|---|
| Arguments: | extentx, extent_y, extenet_z, voxel side length |
| Returns: | Vector of Voxels with given .poly and empty .pass, .stop |
| Description: | Creates a 3D environment of Voxels within a given extent and a given resolution. |

Function **intersects()**

| | |
|---|---|
| Arguments: | Meshes.Box, Meshes.Segment |
| Returns: | Bool (0/1) |
| Description: | Intersects a Voxel (Meshes.Box) with a laser ray (Meshes.Segment) using the Möller–Trumbore intersection algorithm |

Function **intersects()**

| | |
|---|---|
| Arguments: | Meshes.Box, Meshes.Point |
| Returns: | Bool (0/1) |
| Description: | Intersects a Voxel (Meshes.Box) with an end point of a laser beam (Meshes.Point) |

Function **ray_voxel_intersect()**

| | |
|---|---|
| Arguments: | Voxel, Meshes.Segment |
| Returns: | adds 0 or 1 to .pass value of a Voxel |
| Description: | **intersects**(Meshes.Box, Meshes.Segment) test is performed, if the ray intersects the Voxel, value of 1 is added to .pass of the tested Voxel. |

Function **stop_voxel_intersect()**

| | |
|---|---|
| Arguments: | Voxel, Meshes.Point |
| Returns: | adds 0 or 1 to .stop value of a Voxel |
| Description: | **intersects**(Meshes.Box, Meshes.Point) test is performed, if the endpoint intersects the Voxel, value of 1 is added to .stop of the tested Voxel. |

Function **raytrace!**

| | |
|---|---|
| Arguments: | Vector of Voxels, Vector of rays, Vector of endpoints |
| Returns: | Vector of analysed Voxels |
| Description: | Loops **ray_voxel_intersect()** and **stop_voxel_intersect()** through the entire system. |

Function **get_middles()**

Arguments: DataFrame

Returns: DataFrame with coordinates of Voxel middles (X, Y, Z)

Description: Calculates the coordinates of the Voxel middles according to their Meshes.Box geometry in the .poly field. Columns “middles_x”, “middles_y” and “middles_z” are added to the DataFrame.

Function **openness()**

Arguments: DataFrame

Returns: Openness quantifier of a Voxel

Description: Calculates the openness quantifier of a given Voxel (or of a vector of Voxels) based on the number of rays passing through the Voxel and ending in the voxel. The formula is as follows:

$$Openness_{Vi} = \frac{(.pass_{Vi} - .stop_{Vi})}{.pass_{Vi}}$$

Function **focus()**

Arguments: DataFrame

Returns: Focus quantifier of a Voxel

Description: Calculates the focus quantifier of a given Voxel (or of a vector of Voxels) based on the number of rays passing through the Voxel to the sum of .pass of the entire environment. The formula is as follows:

$$Focus_{Vi} = \frac{.pass_{Vi}}{\sum .pass_{V}}$$

Function **occlusion()**

Arguments: DataFrame

Returns: Occlusion of the Voxel (0/1)

Description: Determines if the Voxel is occluded based on the number of rays passing through the Voxel. If there are no rays traversing the Voxel, it is considered occluded (1). If there are rays traversing it, the Voxel was not occluded (0).

Function **rt_quantifiers()**

| | |
|---|---|
| Arguments: | DataFrame |
| Returns: | DataFrame with added raytracing quantifiers – openness, focus, occlusion |
| Description: | Calculates the raytracing quantifiers of openness, user focus and occlusion and adds them in separate columns to the raytraced DataFrame. |

Function **filter_underground_occ()**

| | |
|---|---|
| Arguments: | DataFrame, voxel resolution, GroundPoints, GroundHeader |
| Returns: | DataFrame containing only the Voxels above presumed ground |
| Description: | Determines if a voxel is above or below ground. If it is below the presumed ground, it is filtered out. |

Function **occlusion_rate()**

| | |
|---|---|
| Arguments: | DataFrame |
| Returns: | percentage (%) of occluded Voxels in the given environment |
| Description: | Calculates the proportion of occluded Voxels to the number of Voxels in the environment, returns percentage (%) value. |

Function **top_vox()**

| | |
|---|---|
| Arguments: | GroupedDataFrame, threshold |
| Returns: | Coordinates of a TopVox (Z, X, Y) in a column |
| Description: | Finds the highest non-occluded and non-open voxel in a given column (GroupedDataFrame). This local TopVox must have at least two laser beams returning from it as a failsafe against outliers and scanner errors. If a voxel is considered open or not is decided according to the given threshold. If the openness value exceeds the threshold, Voxel is open. If the openness is equal or lower than threshold, it is not open. |

Function **get_topvox()**

Arguments: DataFrame, threshold

Returns: Coordinates of a global TopVox (Z, X, Y)

Description: Cycles the **top_vox()** functions through all groups in a GroupedDataFrame and finds the coordinates of the global TopVox.

Function **gt_height()**

Arguments: DataFrame, threshold, GroundPoints, GroundHeader

Returns: G-T height

Description: Using the **get_topvox()** finds the TopVox coordinates and measures the distance between its centre and the ground perpendicularly below it. The Z value of the ground is taken as a mean value of a 2x2 metre square with the TopVox X and Y coordinates in its middle.

Function **voxel_viz_openness()**

Arguments: Voxel DataFrame, solid, threshold

Returns: GLMakie Scene

Description: Visualization of the openness, colour and alpha represent openness. If solid = true, will visualize only non-open voxels, given by the threshold argument; if solid = false, will visualize all voxels.

Function **voxel_viz_focus()**

Arguments: Voxel DataFrame

Returns: GLMakie Scene

Description: Visualization of the user focus, colour and alpha represent the focus quantifier. To better visualize the user focus quantifier, it is exaggerated by a function:

$$100 * \sqrt{focus_{Vi}}$$

Function **voxel_viz_solids()**

Arguments: Voxel DataFrame, clr, threshold

Returns: GLMakie Scene

Description: Visualization of non-open voxels, colour of voxels is given by the clr argument, non-openness is set by the threshold argument, alpha is set to 0.5.

Function **voxel_viz_occlusion()**

| | |
|---|---|
| Arguments: | Voxel DataFrame, clr |
| Returns: | GLMakie Scene |
| Description: | Visualization of occluded voxels within the environment, colour of voxels given by the clr argument, alpha is set to 0,5. |

## Publishing and archiving

Approved students' theses at SLU are published electronically. As a student, you have the copyright to your own work and need to approve the electronic publishing. If you check the box for **YES**, the full text (pdf file) and metadata will be visible and searchable online. If you check the box for **NO**, only the metadata and the abstract will be visible and searchable online. Nevertheless, when the document is uploaded it will still be archived as a digital file. If you are more than one author, the checked box will be applied to all authors. You will find a link to SLU's publishing agreement here:

- https://libanswers.slu.se/en/faq/228318.

☒ YES, I/we hereby give permission to publish the present thesis in accordance with the SLU agreement regarding the transfer of the right to publish a work.

☐ NO, I/we do not give permission to publish the present work. The work will still be archived and its metadata and abstract will be visible and searchable.